VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2012 Bc. Michal Dojcsán

VŠB – Technická univerzita Ostrava
Fakulta elektroniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Řízení zpětnovazebního systému založené na NCS
Feedback system control based on NCS

2012 Bc. Michal Dojcsán

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Michal Dojcsán**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Řízení zpětnovazebního systému založené na NCS
Feedback System Control Based on NCS

Zásady pro vypracování:

Diplomová práce se zabývá realizací zpětnovazebního systému na základě teorie Networked Control Systems a jeho regulací na vybrané laboratorní úloze. V souhrnu je práce charakterizována těmito body:

1. Obecný popis problematiky NCS.
2. Analýza a návrh laboratorní úlohy NCS s ohledem na možnosti vzdáleného řízení.
3. Rešerše možných řešení z hlediska operačního systému.
4. Sestavení základního regulačního řetězce.
5. Testy kritických parametrů pro regulaci.
6. Implementace části měření a akčního zásahu pomocí vhodné karty pro sběr dat (DAQ).
7. Implementace regulátoru a vizualizace úlohy v PC.
8. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

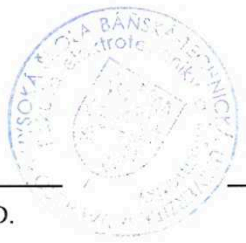
[1] GREPL, R. *Modelování mechatronických systémů v Matlab/SimMechanics.* Praha: BEN-technická literatura, 2007. 152 s. ISBN 978-80-7300-226-8.
[2] *Control Engineering, including: Fuzzy Control System, Pid Controller, Inverted Pendulum, Distributed Control System, Control System, Life-critical ... (automatic Control), Sampled Data Systems.* Hephaestus Books, 2011. 170 s. ISBN 978-1243352958.
[3] BUBNICKI, Z. *Modern Control Theory.* Springer, 2010. 423 s. ISBN 978-3642063015.
[4] WANG, F. - LIU, D. *Networked Control Systems: Theory and Applications.* Springer, 2010. 344 s. ISBN 978-1849967563.
[5] ALUR, R. - ARZEN, K.-E. - BAILLIEUL, J. - HENZIGER, T.A. - HRISTU-VARSAKELIS, D. - LEVINE, W.S. *Handbook of Networked and Embedded Control Systems (Control Engineering).* Birkhäuser Boston, 2008. 822 s. ISBN 978-0-8176-3239-7.
[6] Oficiální dokumentace RTAI.
[7] Oficiální dokumentace RTX/Windows.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Zdeněk Slanina, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

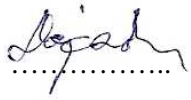


doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení:

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.



Datum odevzdání diplomové práce: 4.5.2012

Děkuji panu Ing. Zdeňkovi Slaninovi, Ph.D. za odbornou konzultaci a pomoc při zpracování diplomové práce, zvláště pak za kompilaci jádra operačního systému Linux a instalací příslušného software.

Abstrakt:

Cílem této diplomové práce je realizace a nastínění různých možností vzdáleného řízení, vzhledem k operačnímu systému, způsobů komunikace na síti Ethernet a vhodného algoritmu pro řízení vybraného modelu. Byl navržen a realizován regulační systém, který tvořili dva účastníci UDP/IP komunikace po lokální Ethernetové síti. Z toho jeden v roli serveru ovládal vybraný model Kulička na tyči od firmy Humusoft, pomocí multifunkční karty MF624 a druhý v roli klienta a regulátoru přijímal zprávy o soustavě a zasílal obratem zpět informace o akčním zásahu, který byl prováděn algoritmem PSD a byl implementován v jazyce C.

Abstract:

The aim of the thesis is the realization and outline of various options for remote control considering operating system, methods of communication on the Ethernet network and an appropriate algorithm for control of the selected model. Regulatory system consisting of two participants of UDP/IP communication on a local Ethernet network was designed and realized. One of the participants in the server role controls chosen model ball on an inclined plane made by Humusoft Company using multifunction card MF624. The other participant in the role of client and regulator receives reports on the system and it sends back information about action that was carried out by PSD algorithm that was implemented in the C language.

Klíčová slova

Ethernet, NCS, kulička na tyči, řízení po síti, vzdálené řízení, RTAI, Xenomai, RTnet, Linux, PSD, Anti Wind Up efekt

Keywords:

Ethernet, NCS, ball on an inclined plane, networked control, remotely control, RTAI, Xenomai, Linux, PSD, Anti Wind Up effect

Seznam zkratek:

A/Č	Analogově číslicový
ACPI	Advanced Configuration and Power Interface
API	Application Programming Interface
ARP	Address Resolution Protocol
CPU	Central Processing Unit
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
DAQ	Data Acquisition
DCS	Decentralized Control Systems
DDK	Driver Development Kit
DMA	Direct memory access
GPL	General Public License
GSM	Global System for Mobile Communication
HAL	Hardware Abstraction Layer
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IRQ	Interrupt Request
LAN	Local Area Network
LED	Light Emitting Diode
NCS	Networked Control System
NIC	Network Interface Card
NTP	Network Time Protocol
OS	Operating System
PC	Personal Computer
PID	Proporcionálně - Integračně - Derivační
POSIX	Portable Operating System Interface
PSD	Proporcionálně - Sumačně – Diferenční
PWM	Pulse - Width Modulation
QoS	Quality of service
RS232	Recommend Standard 232
RTAI	Real-Time Linux Application Interface

RTOS	Real-Time Operating System
RTSS	Real-Time Systems Symposium
RTX	Real-Time eXtension
SMI	System Management Interrupt
SMP	Symmetric Multi-Processing
SMS	Short Message Service
SNT	Simple Network Time Protocol
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
ZOH	Zero Order Hold

Seznam symbolů

K_d	Derivační regulační konstanta
K_i	Integrační regulační konstanta
K_p	Proporcionální regulační konstanta
F_n	Normálová síla
F_g	Tíhová síla
F_p	Potencionální síla
E_k	Kinetická energie
E_p	Potencionální energie
J_k	Moment setrvačnosti
m	Hmotnost
W	Práce
G_s	Přenosová funkce v Laplaceově transformaci
x	Poloha
$\dot{x}$	Rychlost
φ	Úhel
$\dot{\varphi}$	Úhlový kmitočet
L	Lagrangián

Obsah:

1 Úvod

Cílem práce je navrhnout a vytvořit pro vybraný model vzdálené řízení, založené na koncepci NCS (Networked Control System). Regulátor má být tedy fyzicky vzdálený od řízeného modelu a akční zásahy vykonávat prostřednictvím sítě.

Práce se zabývá různými řešeními z hlediska operačních systémů pracujících v reálném čase a dále pak staví na koncepci operačního systému GNU/Linux s příslušným RT rozšířením Xenomai a komunikaci protokolem UDP/IP k dosažení vhodných časový parametru pro řízení vybraného modelu kulička na tyči od firmy Humusoft.

Dále se práce zabývá návrhem, simulací a následnou implementací vybraného algoritmu pro řízení modelu, který je vhodný jak ze strany charakteru soustavy, tak ze strany různých omezení plynoucích z koncepce vzdáleného řízení po síti Ethernet.

Vytvořené programy využívající koncepci producent-konzument dosahují na síti LAN dostatečně rychlé komunikace pro zavedení regulační smyčky, která dokáže plynule regulovat kuličku do žádané polohy. Jednoduchá vizualizace pohybující se kuličky v příkazovém řádku společně v kombinaci s náhledem na číselná data o stavu soustavy přibližují pozorovateli aktuální chování modelu, který může být např. v jiné místnosti.

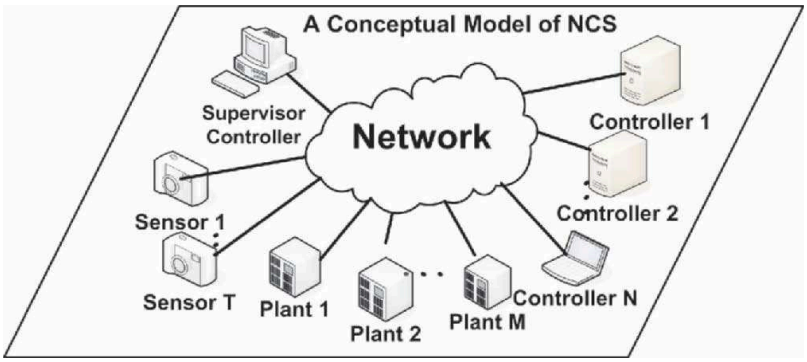
Samotná regulace, která je výsledkem PSD algoritmu implementována v jazyce C společně s vhodnými doplňky, jako jsou Anti-Wind-Up efekt, změna konstant nebo filtrace vstupní veličiny je do určitých změn žádané veličiny bez překmitu a s úplným ustálením pozice kuličky.

2 Obecný popis problematiky NCS

2.1 Základní definice NCS

Po mnoho let nám vědci poskytují přesné a optimální strategie řízení systémů vznikající z klasické teorie řízení, a to od otevřené smyčky až po složité strategie řízení na základě genetických algoritmů. Příchod komunikačních sítí představil koncept dálkově ovládaných systémů, který způsobil zrod Networked Contol System, dále už pouze NCS.

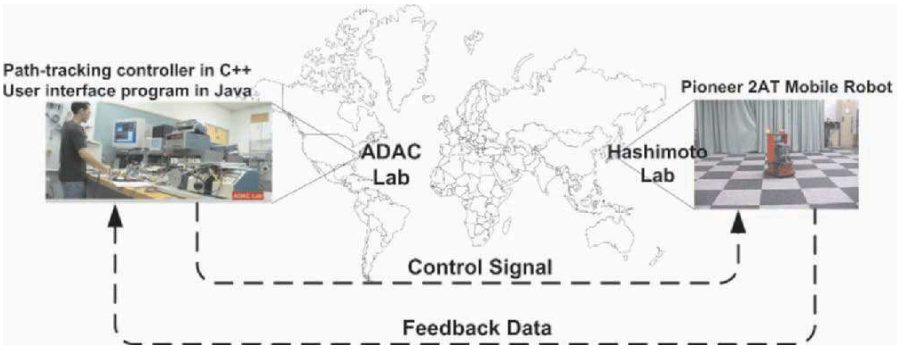
Klasickou definici NCS je možno uvést takto: Pokud je tradiční zpětnovazební řízení uzavřeno pomocí komunikačních kanálů, které mohou být sdíleny s dalšími uzly mimo samotné řízení systému, pak je řízení nazýváno jako NCS. Dále jde NCS definovat jako zpětnovazební řízení, jehož smyčka je uzavřena prostřednictvím sítě v reálném čase. Charakteristickým znakem je to, že NCS informace (referenční vstupy, výstupy zařízeni, vstupy k řízení, atd.) se vyměňují za použití sítě mezi řídicím systémem a komponenty (senzory, pohony, atd., viz Obr.2.1).
[1]



Obr. 2.1 .Typický NCS systém[1]

2.2 Komunikace u NCS:

Komunikační kanál je páteří NCS, spolehlivost bezpečnost, snadnost použití a dostupnost jsou hlavním cílem při výběru komunikace nebo typu přenosu dat. V dnešním světě je spousta možností, k dispozici jsou telefonní linky, sítě mobilních telefonů, satelitních sítí a nejpoužívanější internet. Ethernet je nejvhodnější a levnou volbou pro mnoho aplikací, kde jsou řízený systém a samotný regulátor daleko od sebe, jak je ukázáno na Obr.2.2.



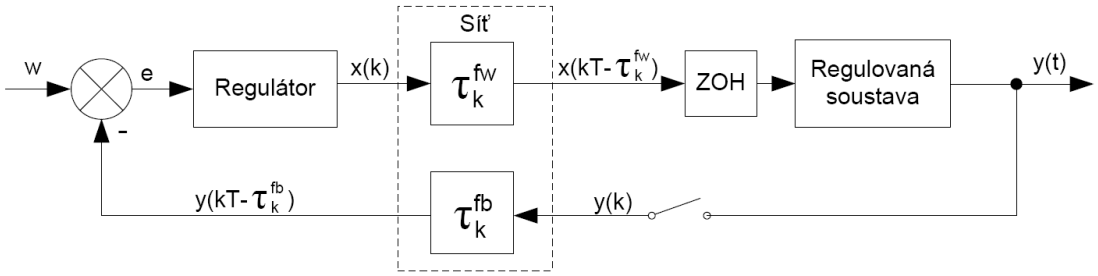
Obr. 2.2 Typický systém vzdáleného řízení

2.3 Výzvy a řešení v NCS problematice:

Po představení a přehledu různých kategorií NCS koncepce je možno se zabývat problémy při návrhu vzdáleného řízení. V zásadě můžeme roztřídit NCS aplikace do dvou kategorií. Časově citlivých aplikací nebo též časově kritických aplikací, jako jsou: vojenství, řízení dopravy, apod. A časově necitlivých nebo ne-Real-time aplikací, jako jsou: ukládání dat, sběr dat ze senzorů, e-mail, apod.
[1]

Nicméně, spolehlivost sítě je důležitým faktorem pro obě tyto kategorie. Síť se může ukázat jako nespolehlivá a časově závislá na úrovni služeb, například v oblastech: zpoždění, časové nestability periody (jitter), nebo ztráty dat. QoS mohou posunout chování sítě až k vlastnostem Real-time co do chování v čase, ale chování sítě bude stále předmětem úprav (zejména v bezdrátových médiích), přechodného směrování a agresivních toků. Na druhé straně sítě můžou rozmary výše zmíněného nedeterminismu ohrozit stabilitu, bezpečnost a výkon jednotky fyzického prostředí. Velký problém v řízení založeném na síti je vliv časového zpoždění v ní. Čas na čtení ze senzoru, měření a poslání řídicích signálů pro pohony přes síť závisí na její charakteristice, jako je topologie a směrování systémů. Proto může být celkový výkon NCS ovlivněn výrazným zpožděním sítě. Závažnost zpoždění problém ještě zhoršuje, pokud nastane během přenosu ke ztrátě dat. Kromě toho, že zpoždění degradují výkon sítě založené na NCS, ale také může destabilizovat řízený systém.
[1]

Při návrhu řídicích algoritmů s využitím NCS koncepce od PID, optimálního, adaptivního a robustního řízení po znalostní a další pokročilé formy řídicích algoritmů je nutno upravit strategii řízení podle požadavků aplikace, tak aby mohli spolehlivě pracovat po síti s kompenzací dopravního zpoždění a nepředvídatelnosti. Obr.2.3 zobrazuje typický NCS model s dopravním zpožděním.
[1]



Obr.2.3 NCS s dopravním zpožděním v síti [1]

W	žádaná veličina
e	Regulační odchylka
x	Akční veličina
y	Výstup soustavy
τ_k^{fw}, τ_k^{fb}	Laplaceovy obrazy časových zpoždění v kroku *k*.
ZOH	Tvarovač nultého řádu

2.4 Využití Ethernetu jako komunikačního standardu pro NCS

Ethernet 802.3 nesplňuje požadavky na práci v reálném čase (Real-time), a tudíž je jeho použitelnost v automatizaci omezena. Z hlediska práce v reálném čase nepřispívají k dobrým vlastnostem Ethernetu metoda CSMA/CD , to je metoda mnohonásobného přístupu s nasloucháním nosné a detekcí kolizí. Tato metoda přístupu k médiu je velmi efektivní při nižším zatížení sítě (cca 30 % šířky pásma). Její efektivita klesá při větším počtu zájemců o vysílání, kdy může dojít k exponenciálnímu nárůstu kolizí. Efektivita CSMA/CD je vyšší pro delší rámce, protože při jejich přenosu je výhodnější poměr mezi trváním kolizního okénka a vysílání dat.
[2], [4]

Zásadní otázkou a nejčastější výtkou na adresu Ehernetu z pohledu jeho použití v NCS je tudíž nedeterminismus Ethernetu TCP/IP. Přitom determinismus komunikačního podsystému decentralizovaného systému řízení DCS, stejně jako všech řídicích systémů, je nezbytným předpokladem pro jejich činnost v reálném čase.
[2], [4]

K posílení vlastností Ethernetu jako systému reálného času se v současné době používají následující metody, zčásti již využívané v Ethernetu TCP/IP pro účely internetu:

2.4.1 Přepínání

Použití rozdělovačů (hubů) ve strukturované kabeláži informačních systémů má za následek, že původní jedna doména, zahrnující v topologii sběrnice jeden celý segment sítě Ethernet, je rozbita, a tím zmenšena na mnohem menší segmenty, v nichž je fyzicky přítomen vlastně jen rozdělovač a cílová stanice. Avšak z hlediska přístupu k síti metodou CSMA/CD rozdělovač obecně nezmenší pravděpodobnost vzniku kolize, a tím neposouvá tuto metodu směrem k charakteristikám systémů reálného času. Jiná situace nastane, použije-li se místo rozdělovače (hub) přepínač (switch) pracující na principu „ulož a pošli“ (store and forward). Přepínač umožní vytvořit velmi malý segment a je zároveň vybaven mechanismy ukládání paketů a jejich směrování jen jednomu cílovému účastníku, a tím posiluje vlastnosti Ethernetu jako systému reálného času směrem k vyššímu stupni determinismu.
[3]

2.4.2 Vysokorychlostní Ethernet

V současné době pracuje většina Ethernetových sítí přenosovou rychlostí 10 nebo 100 Mb/s. Rychle se však rozšiřuje použití sítí Ethernet s přenosovými rychlostmi 1 Gb/s, popř. 10 Gb/s, a je dokončena standardizace sítě s rychlostí přenosu 100 Gb/s. Není pochyb o tom, že vývoj k větším rychlostem půjde dál. S každým desetinásobným nárůstem rychlosti sítě se zkrátí doba potřebná na přenos jednoho paketu na desetinu. Při přenosové rychlosti 10 Mb/s trvá přenos Ethernetového paketu délky 1 522 bajtů asi 1,2 ms. Použije- li se rychlý Ethernet, je potřebná doba jen asi 120 μs, atd. Zkracování doby potřebné pro přenos zpráv logicky podporuje v chování systémů vlastnosti reálného času.
[3]

2.4.3 Plně duplexní přenos

Je-li součástí strukturované kabeláže místo koaxiálního kabelu několika-žilový kabel, lze snadno použít úplný duplexní přenos. Stanice mohou současně zprávy vysílat i přijímat, čímž vzroste rychlost obousměrné komunikace dvakrát. Zmenšuje se také pravděpodobnost kolize. V současnosti je už velmi nepravděpodobné setkání s poloviční duplexem, neboť pro většinu instalací se využívají nestíněné kabely s kroucenými páry vodičů UTP s plným duplexem.
[3]

2.4.4 UDP místo TCP

Protokol TCP je orientován na ustavení virtuálního spojení mezi vysílací a přijímací stanicí (connection oriented) na začátku přenosu a ukončení virtuálního spojení po ukončení relace. Zajišťuje tedy potvrzování a přeposílání předávaných zpráv. V důsledku toho se pozná, zda předávaná data byla doručena. Jestliže nebyla, jsou automaticky poslána znovu. Protokol TCP dokonce zaručuje, že předávané pakety dojdou v pořadí, v jakém byly odvysílány. Protokol UDP naopak představuje nespojovanou službu (connectionless oriented). Poslaná data jsou navzájem zcela nezávislá.
[3]

Protokol UDP je protokol transportní vrstvy. Stejně jako TCP slouží ke komunikaci dvou aplikací. Stejně jako TCP používá IP jako síťový protokol (pro spojení dvou počítačů). Stejně jako TCP identifikuje aplikace na počítačích pomocí tak zvaného portu (číslo). UDP port je jednoznačné číslo identifikující aplikaci. Na jednom počítači nemohou dvě aplikace používat stejný UDP port. Čísla TCP portů a UDP portů jsou na sobě nezávislá. Jeden program může používat například TCP port 5000 a jiný na stejném počítači může používat UDP port 5000. Podstatným rozdílem je, že UDP není spojová služba. Tedy nenavazuje se spojení. Co z toho vyplývá? (krom toho, že se nemusí volat *connect*)

- Není potvrzováno doručení UDP data-gramů - data pošleme a tím nad nimi ztrácíme jakoukoliv kontrolu. Možná druhé straně dojdou, možná ne.
- Neexistuje kontrolní součet - data se (teoreticky) mohou poškodit.
- Data mohou být doručena ve špatném pořadí. Tedy data1 a data2 (z předchozího odstavce) mohou být doručená v pořadí data2 a potom data1. Nejsme schopni to nijak ovlivnit. Samozřejmě, že jednotlivá byte v blocích dat (data1 a data2) nemohou být přeházená.
- Strana, která přijímá, je schopná rozlišit jednotlivé data-gramy. Tedy data odeslána jedním *send* (resp. *sendto*) jsou přijata jedním *recv* (resp. *recvfrom*). Ale není garantováno, že budou ve stejném pořadí, mohou se předbíhat.
- Data se mohou duplikovat.

Výhody UDP:

- Malá hlavička dat
- Malé zatížení sítě (neposílá se potvrzování)
- Při přijímání lze rozlišit jednotlivé diagramy

Nevýhody UDP:

- Naprosto nezabezpečený přenos (data se mohou ztratit, poškodit, duplikovat, předbíhat. Nic z toho nejsme schopni zjistit.)

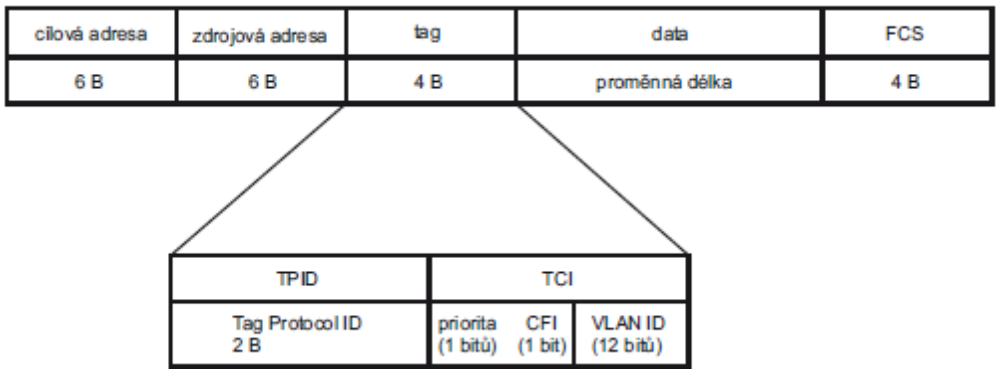
Protokol UDP nezná potvrzování zpráv ani ustavení a ukončení spojení. Proto při chybě při přenosu mohou být protokolem UDP přenesena data znovu, a to hned v dalším cyklu, což u protokolu TCP není možné. Protokol TCP při poruše opakuje přenos dat tak dlouho, dokud nejsou správně přijata. Protokol UDP je proto mnohem „rychlejší“ a jednodušší protokol čtvrté vrstvy modelu ISO/OSI než TCP, a proto se mu v průmyslových variantách Ethernetu dává přednost. Využití UDP se také nabízí při různých " Real -time" přenosech multimediálních dat. Například různé Internetové videokonference. Přenáší se velký objem dat a jejich potvrzování by bylo pro síť opravdu náročné. Stejně tak rychlá komunikace v malé časové periodě by potvrzování nebo kontrola dat mohla příliš zpomalit.
[21]

2.4.5 Metody typu producent-konzument a publisher-subscriber

Metody navazování, udržování a ukončování spojení v sítích Ethernet TCP/IP podle modelů producent-konzument nebo publisher-subscriber, které se již v informatice používají, jsou pro dosahování vyššího stupně determinismu v Ethernetových sítích vhodnější než způsob klient-server. V obou případech jde o multicasting, tedy posílání zpráv většímu počtu účastníků současně. Smyslem uvedených metod je sloučit příjemce určitých dat do skupin. V případě modelu publisher-subscriber se zájemci (subscribers) o příjem dat registrují u poskytovatelů těchto dat. S využitím tabulek příjemců (subscribers) potom poskytovatelé (publishers) data rozesílají. V případě modelu producent-konzument nejsou udržovány žádné tabulky zájemců. Data jsou místo toho opatřena identifikátorem, na základě kterého příjemci (konzumenta) odebírají zprávy od poskytovatele (producenta).
[21]

2.4.6 Prioritní sloty v protokolu sítě Ethernet

Dalším mechanismem, s nímž lze dosáhnout vyššího stupně determinismu některých časově kritických zpráv při použití metody CSMA/CD, je využití tzv. prioritních slotů na úrovni druhé vrstvy podle standardu IEEE 802.1p.

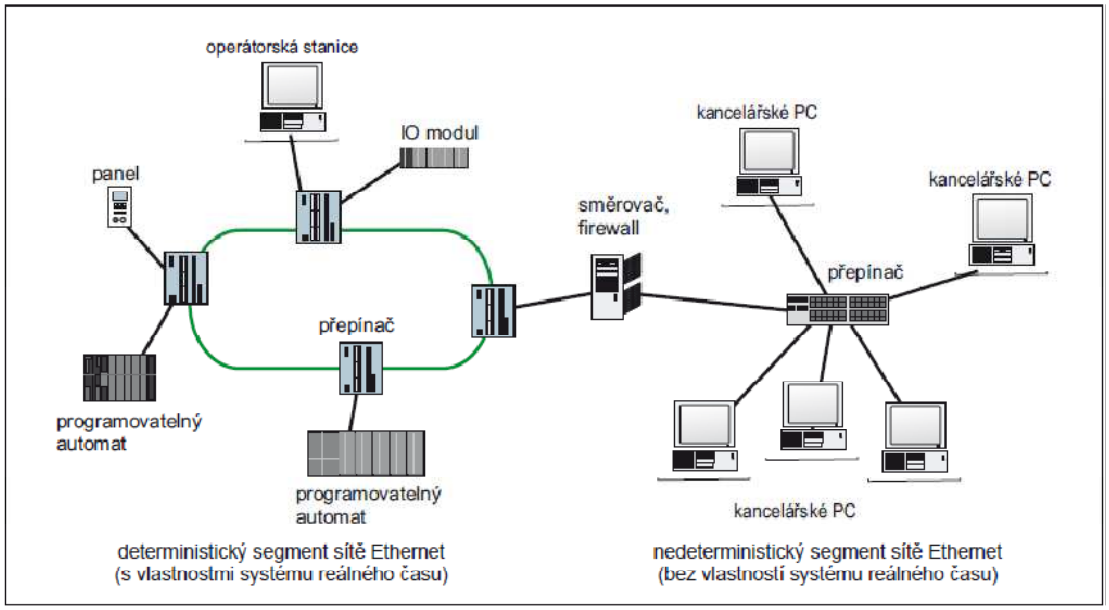


Obr. 2.4.6 Prioritní sloty ve druhé vrstvě protokolu Ethernet (IEEE 802.1p) [3]

Při použití metody prioritních slotů je uvnitř rámce protokolu spojové vrstvy v síti Ethernet umístěno pole s názvem *tag*, nesoucí informaci o prioritě předávané zprávy (obr. 2.4.6). Zprávám s nejvyšší prioritou jsou vyhrazeny první prioritní sloty. Přepínače podporují nárůst determinismu sítě tím, že segmenty sítě (kanály) jsou rozděleny podle priorit. Časově kritické zprávy jsou pak přepínači směrovány do příslušných segmentů (kanálů) podle úrovně priority. [3]

2.4.7 Segmentace sítě na deterministické a ostatní části

Protože síť průmyslový Ethernet se používá a bude používat nejen pro přenos technologických dat, ale i pro vizualizaci, sledování na dálku přes internet, ovládání prostřednictvím e-mailových zpráv a zpráv SMS i dalších služeb, je efektivní oddělit zprávy vyžadující přenos v reálném čase od ostatních zpráv, které striktně deterministický přenos nevyžadují. Proto jsou zařízení komunikující v reálném čase spojována přes co nejmenší počet přepínačů, neboť čím větší počet přepínačů je v cestě signálu, tím větší je jeho zpoždění. Možný způsob koexistence Ethernetových sítí reálného času (deterministických) a sítí charakteru „bez“ reálného času (nedeterministických) je naznačen na obr. 2.4.7.



Obr. 2.4.7. Segmentování sítě Ethernet

Zvlášť náročnou částí návrhu takové smíšené sítě je projekt oddělení segmentů s převážně internetovým (nedeterministickým) provozem a segmentů s provozem převážně deterministickým. Rozhraní mezi těmito dvěma druhy provozu musí být přesně řízena přepínačem. Aby se předešlo přetížení segmentu s deterministickým provozem, musí být limitováno množství dat, která do něj vstupují. Nedeterministická část sítě bude např. používat rychlost 10 Mb/s a deterministická 100 Mb/s (a obdobně u větších rychlostí). Jiný způsob, jak oddělit deterministický provoz od nedeterministického, spočívá v použití směrovačů a firewallů.

Faktorem ovlivňujícím zatížení sítě je také počet zpráv typu broadcast v síti. Zatěžována jsou jak koncová zařízení (musí vyhodnotit každou zprávu typu broadcast), tak i přepínače (protože přepínač musí kopírovat každou zprávu typu broadcast na všechny výstupní porty). U některých přepínačů lze proto limitovat počet zpráv typu broadcast zpracovaných např. za sekundu.

2.4.8 Synchronizace v Ethernetových sítích

K řešení široké škály deterministických úloh přenosu dat postačují dosud uvedené mechanismy, které dokážou zajistit dodržení požadované doby odezvy sítě (Deadline). Také se již ve značné míře používají jak v Ethernetových sítích pro oblast informačních systémů, tak zejména v různých průmyslových variantách Ethernetu (Industrial Ethernet). Druhé vlastnosti systémů reálného, časové nestability – však současnými variantami Ethernetu TCP/UDP/IP nelze dosáhnout pro poměrně velkou třídu deterministických úloh. A to ani při použití již uvedených mechanismů. Typickými příklady takových úloh je řízení pohonů (polohy a pohybu) a realizace bezpečných systémů.

[3]

Průmyslové sběrnice, které jsou vyvinuty speciálně pro systémy reálného času, řeší otázku synchronizace tak, že řídicí systém sběrnice jednoznačně plánuje čas předávání zpráv jednotlivými účastníky. Časový rozvrh komunikace a časový rozvrh provedení akcí si jsou jednoznačně přiřazeny. Systém je deterministický, tj. lze přesně určit okamžik provedení akcí. Při řízení sběrnice způsobem master-slave, používaným u průmyslových sběrnic kategorie fieldbus, je zcela pod kontrolou synchronní chod komunikujících řídicích systémů (např. řídicích jednotek dvou a více pohonů os obráběcích strojů). Naproti tomu protokol Ethernet v principu tuto synchronizaci nezajišťuje (v důsledku použití přístupové metody CSMA/CD). Časový rozvrh komunikace a provedení akcí si nejsou jednoznačně přiřazeny. Systém je zatížen velkým tolerančním pásmem (jitter) okolo požadovaného okamžiku provedení akce (Deadline nebo též časové uzávěry), a je tudíž nedeterministický.
[3]

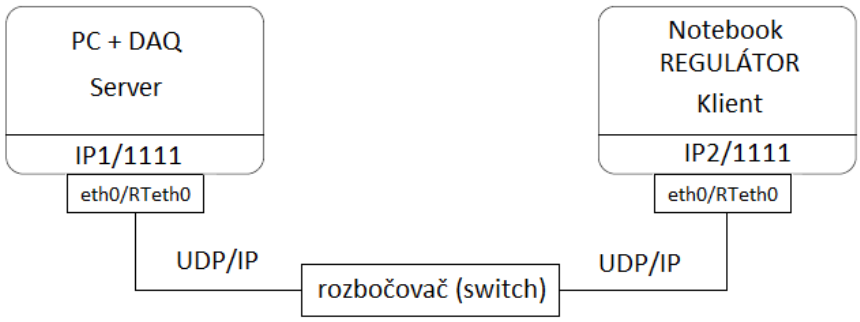
Synchronizační mechanismy použité v sítích LAN, např. NTP a SNTP, uspokojivě neřeší požadavky průmyslové automatizace, třebaže je někteří dodavatelé „průmyslového“ Ethernetu zavedli. Pro účely automatizace je třeba do sítě Ethernet TCP/IP pro jejich přizpůsobení pro práci v reálném čase zavést levný synchronizační prostředek, takový, který nebude mít velké dodatečné požadavky na výkonnost jednotlivých účastníků sítě.

Jedním z nových způsobů převodu sítě Ethernet TCP/IP na deterministický systém je metoda podle standardu IEEE 1588 s označením PTP (Precision Time Protocol). Princip této metody, který spočívá v synchronizaci účastníků sítě prostřednictvím distribuovaných hodin reálného času, je použitelný v libovolné síti, do níž může vstupovat větší počet uživatelů. Z našeho pohledu tedy především v lokálních sítích Ethernet TCP/IP. Praxe ukázala, že metoda PTP umožňuje Ethernetu TCP/IP dosáhnout lepší synchronizace, než jaké dosahují současné sběrnice typu fieldbus. Proto se tato metoda používá v současných variantách průmyslového Ethernetu určených pro časově zvlášť kritické úlohy (v režimu tzv. Hard Real-time). Jde o elektrické pohony (synchronizace řízených os), systémy distribuce energie (korelování dat naměřených v prostorově distribuované energetické síti), systémy zálohující jiné informační kanály (ztráta spojení prostřednictvím GSM), bezpečné systémy a další.
[3]

3 Analýza a návrh laboratorní úlohy NCS.

Úloha NCS regulace bude uskutečněna na reálném modelu kuličky na tyči od firmy Humusoft. K vzdálenému řízení se bude přistupovat jako k předloze server - klient nebo přesnění producent - konzument, jak je naznačeno na Obr. 2.

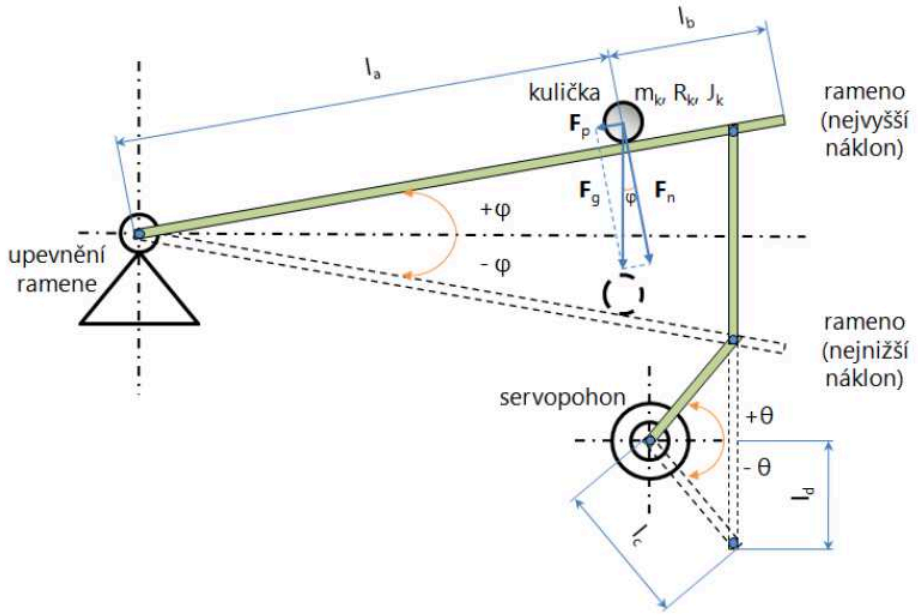
K řízení modelu kuličky na tyči se nabízí desktopové řešení počítače s architekturou x86 (I32), které obsahuje náležitá hardwarová rozhraní pro splnění koncepce řízení po síti NCS, jako je sběrnice PCI pro DAQ měřicí kartu a Ethernetový port pro připojení do sítě LAN nebo Internetu. Je však nutné si dávat pozor a na určité limity, které jsou dány operačním systémem a použitým hardwarem v PC. Způsoby jakými se v těchto platformách vykonávají systémová volání, obsluhy přerušení a funkce typu SMI, DMA, ACPI, atd. mohou mít za následek nevhodné chování řídicího systému jako takového.



Obr. 3 Topologické schéma NCS úlohy, server-klient

3.1 Popis modelu

Jedná se o fyzicky vyrobený model firmou Humusoft, který obsahuje kuličku pohybující se na nakloněné rovině. Hlavní úlohu tedy hraje kulička, která je umístěna na kolejničce. Kolejničku tvoří dvě rovnoběžně napnutá ocelová lanka (modelově je to tyč), která jsou na jednom konci upevněná ve statickém bodě. Na druhém konci jsou lanka upevněná na sloup, který ve vertikální poloze vychyluje nahoru a dolů servomechanismus. Pootočením osy servomotoru, se mění poloha vychylovací tyče a tím se také mění náklon roviny kolejničky. Při změně úhlu, resp. náklonu roviny kolejničky, se vlivem tíhové síly země kulička pohybuje. (viz. Obr. 3.1). [5]



Obr. 3.1 Schéma fyzikálního modelu kuličky na tyči [5]

3.2 Fyzikální popis modelu

Na Obr. 2.1 lze vidět fyzikální schéma modelu. Délka ramene je definována součtem vzdáleností la a l_b, kde l_b je požadovaná pozice kuličky. Kulička je definovaná hmotným středem, jenž má hmotnost m_k a moment setrvačnosti J_k. Poloměr kuličky je označen R_k. Při změně úhlu θ osy servopohonu se pootočí vychylovací tyč, která je připevněna kloubem k rameni modelu. Tímto se rameno naklání o úhel φ. Výška zdvihu opory při otáčení servopohonu je označena vzdáleností l_d a poloměr osy otáčení servopohonu je označen jako vzdálenost l_c. Při naklonění ramene v úhlu $\varphi = 0°$ působí na kuličku gravitační síla F_g. Na tuto sílu působí vlivem podlaží v opačném směru normálová síla F_n. Tyto síly se navzájem odečítají a jejich nulová výslednice ponechává kuličku v klidu. Pokud se rameno nakloní o úhel $\varphi \neq 0°$, pozmění se směr vektoru síly F_n o tentýž úhel. Rozvorem vektorů sil F_g a F_n vzniká výsledný vektor potenciální síly F_p, která pohybuje kuličkou. Model lze rozdělit do dvou důležitých částí:

Mechanická část – Obsahuje všechny mechanické komponenty jako např. ocelovou kuličku, kolejnici z ocelových lanek, rameno se statickým upevněním na jednom konci, pomocnou zdvíhací oporu upevněnou na hřídeli servopohonu a kovový kryt modelu.elektrickou část – El. část zahrnuje veškerou řídicí elektroniku na modelu. Tj. analogový stup a výstup pro řídicí systém, obvody pro vyhodnocování polohy ramene, obvody pro vyhodnocování polohy kuličky, obvod řízení celého modelu, indikace chodu modelu, napájecí zdroj pro celý model s přístrojovou svorkovnicí. [5]

Elektrickou část – El. část zahrnuje veškerou řídicí elektroniku na modelu. Tj. analogový stup a výstup pro řídicí systém, obvody pro vyhodnocování polohy ramene, obvody pro vyhodnocování polohy kuličky, obvod řízení celého modelu, indikace chodu modelu, napájecí zdroj pro celý model s přístrojovou svorkovnicí. [5]

Model (resp. zdroj modelu) je napájen síťovým napájením 230 [VAC]. Součástí napájecího modulu je tavná pojistka. Vypínač je realizován páčkovým spínačem. Zelená LED dioda indikuje zapnutý stav. Vstupy a výstupy jsou realizované komunikačním rozhraním RS232 s devíti pinovým konektorem CANON9. Tyto analogové vstupy a výstupy pracují v napěťových rozsazích 0 [VDC] až 10 [VDC]. Napětí analogového vstupu je akční zásah pro servopohon. Hodnota napětí analogového výstupu reprezentuje pozici kuličky. Hodnoty jednotlivých fyzikálních veličin popisujících ocelovou kuličku, polohu kuličky, polohu ramene a pomocnou zdvihací oporu se servopohenem se nachází v tabulce. (viz. Tab. 3.2a a Tab. 3.2b) Tab. 3.2a Hodnoty fyzikálních veličin kuličky Tab. 3.2b Hodnoty jednotlivých fyzikálních veličin popisujících polohu kuličky, polohu ramene a pomocnou zdvihací oporu se servopohonem 2. [5]

Poloha kuličky			
Označení veličiny	Hodnota	Jednotka	Význam
$\boldsymbol{l}$	$l = 0{,}95$	$[m]$	aktivní délka ramene
$\boldsymbol{l_b}$	$l_b = l - l_a$	$[m]$	požadovaná vzdálenost kuličky
$\boldsymbol{l_a}$	$l_a = l - l_b$	$[m]$	vzdálenost středu kuličky od pevného konce ramene
$\boldsymbol{g}$	$g - 9{,}80665$	$[m \cdot s^{-2}]$	tíhové zrychlení normální
$\boldsymbol{F_g = G}$	$F_g = m_k \cdot g = 0{,}550$	$[N]$	tíhová síla
$\boldsymbol{F_p}$	$F_p = F_g \cdot sin(\varphi)$	$[N]$	potenciální síla kuličky
$\boldsymbol{F_n}$	$F_n = F_g \cdot cos(\varphi) + m \cdot \ddot{\varphi} \cdot l_b$	$[N]$	normálová síla kuličky
Poloha ramene			
Označení veličiny	Hodnota	Jednotka	Význam
$\boldsymbol{\varphi}$	$\varphi \in \langle -0{,}01;\ +0{,}01 \rangle$	$[rad]$	interval úhlu sklonu ramene
Pomocná zdvihací opora se servopohonem			
Označení veličiny	Hodnota	Jednotka	Význam
$\boldsymbol{l_c}$	$l_c = 0{,}019$	$[m]$	poloměr osy otáčení servopohonu
$\boldsymbol{l_d}$	$l_d = \lvert\varphi\rvert \cdot l = 0{,}095$	$[m]$	délka zdvihu opory při otáčení servopohonu nahoru nebo dolů
$\boldsymbol{\theta}$	$\theta \in \langle -\frac{\pi}{6};\ +\frac{\pi}{6} \rangle$	$[rad]$	interval úhlu natočení servopohonu

Tab. 3.2b Hodnoty jednotlivých fyzikálních veličin. Polohy ramene a pomocnou zdvihací oporu se servopohonem [5]

Ocelová kulička			
Označení veličiny	Hodnota	Jednotka	Význam
$\boldsymbol{m_k}$	$m_k = 0{,}056$	$[kg]$	hmotnost kuličky
$\boldsymbol{R_k}$	$R_k = 0{,}012$	$[m]$	poloměr kuličky
$\boldsymbol{J_k}$	$J_k = \frac{2}{5} \cdot m_k \cdot R_k^2 = 3{,}2256 \cdot 10^{-6}$	$[kg \cdot m^2]$	moment setrvačnosti kuličky
$\boldsymbol{V_k}$	$V_k = \frac{4}{3} \cdot \pi \cdot R_k^3 = 7{,}2382 \cdot 10^{-6}$	$[m^3]$	objem kuličky
$\boldsymbol{\rho_k}$	$\rho_k = 7700$	$[kg \cdot m^{-3}]$	hustota kuličky

Tab. 3.2a Hodnoty fyzikálních veličin kuličky [5]

3.3 Matematický popis modelu

Základ matematického popisu modelu tvoří lagrangeovské pojetí mechaniky ve formě Lagrangeových rovnic II. druhu, umožňujících vytvoření pohybových rovnic soustavy hmotných bodů zavedením tzv. zobecněných souřadnic.
[5]

Nejprve je nutno nalézt matematické relace pro kinetickou energii E_k a potenciální energii E_p soustavy kuličky na tyči. Ve výsledné kinetické energii soustavy uvažujeme čtyři druhy kinetické energie – dva druhy pro kuličku (translace, rotace) a dva druhy pro rameno, respektive tyč. [5]

Pro kuličku platí následující relace dle translační energie kuličky a rotační energie kuličky

$$E_{k1} = \frac{1}{2} m_k \dot{x}^2(t) + \frac{1}{2} \cdot \frac{J_k}{R_k{}^2} x^2(t) = \frac{1}{2} \dot{x}^2(t) \left(m_k + \frac{J_k}{R_k{}^2} \right) \tag{1}$$

[5]

Pro tyč bez kuličky a s kuličkou platí následující relace, tedy kinetická energie tyče a kinetická energie tyče s kuličkou:

$$E_{k2} = \frac{1}{2} \cdot J_t \cdot \dot{\varphi}^2(t) + \frac{1}{2} \cdot m_k \cdot x^2(t) \cdot \dot{\varphi}^2(t) = \frac{1}{2} \dot{\varphi}^2(t) \cdot \left(J_t + m_k x^2(t) \right) \tag{2}$$

[5]

Výsledná kinetická energie soustavy E_k je dána součtem kinetických energií kuličky a kinetických energií tyče bez kuličky a s kuličkou (2), tedy:

$$E_k = E_{k1} + E_{k2} = \frac{1}{2}\dot{x}^2(t)\left(m_k + \frac{J_k}{R_k^{\,2}}\right) + \frac{1}{2}\dot{\varphi}^2(t)\cdot\left(J_t + m_k x^2(t)\right) \tag{3}$$

[5]

kde:

$x(t) \equiv l_b(t)$	okamžitá poloha kuličky na tyči	[m]
$\dot{x}(t) \equiv \dot{l}_b(t)$	okamžitá rychlost kuličky na tyči	[m . s^{-1}]
$\varphi(t)$	okamžitý úhel (fáze) sklonu tyč	[rad]
$\dot{\varphi}(t)$	okamžitý úhlový kmitočet tyče	[rad . s^{-1}]
J_t	moment setrvačnosti tyče (ramene)	[kg . m^2]

Ve výsledné potenciální energii soustavy uvažujeme potenciální sílu kuličky F_p pohybující se po nakloněné rovině ve formě nakloněné tyče (viz obr. 2.1), přičemž platí následující relace, tedy:

$$F_p = F_g \sin(\varphi) = G\sin(\varphi) = k_k g \sin(\varphi) \tag{4}$$

Výsledná potenciální energie soustavy *Ep* je dána relací, tedy:

$$E_p = F_p x(t) = m_k g x(t) \sin(\varphi) \tag{5}$$

[5]

V dalším kroku je nutno určit práci nekonzervativních sil $W(t) \equiv W$, tj. Lagrangeova rovnice II. druhu (v Eulerově-Lagrangeově tvaru) nebude homogenní čili rovna nule a bude reprezentována tzv. zobecněnou potenciálovou funkcí:

$$U[\dot{x}(t), x(t), t],$$

Respektive:

$$U[\dot{\varphi}(t), \varphi(t), t],$$

Přičemž zobecněnými souřadnicemi jsou okamžitá poloha kuličky x(t) a okamžitý úhel sklonu tyče φ (t). Příslušné parciální derivace práce dle polohy kuličky jsou dány relacemi, tedy:

$$\frac{\partial}{\partial x(t)} = \{W(t)\} = -bx(t) - \frac{\xi}{R_k} F_n sign[\dot{x}(t)] \tag{6}$$

parciální derivace práce dle úhlu sklonu tyče:

$$\frac{\partial}{\partial \varphi(t)} \{W(t)\} = F_s l \cos(\varphi) \tag{7}$$

Kde:

b	odpor prostředí při pohybu kuličky	[kg . s−1]
ξ	rameno valivého odporu	[m]
F_s	F síla převodu servopohonu	[N]

[5]

Nyní sestavíme obecný předpis k předešlým rovnicím (6) a (7) levé strany ve tvaru Lagrangeovy rovnice II. druhu pro nekonzervativní systém, tedy:

$$\frac{d}{dt}\left\{\frac{\partial}{\partial \dot{q}_i(t)} L(t)\right\} - \frac{\partial}{\partial q_i(t)} \{L(t)\} = \frac{\partial}{\partial q_i} \{W(t)\} \tag{8}$$

Kde:

$L(t) \equiv L = E_k - E_p$	lagrangián	[J]
$q_{i=1}(t) = x(t)$	okamžitá poloha kuličky	[m]
$\dot{q}_{i=1}(t) = \dot{x}(t)$	okamžitá rychlost kuličky na tyči	[m . s^{-1}]
$q_{i=2}(t) = \dot{\varphi}(t)$	okamžitý úhel (fáze) sklonu tyče	[rad]
$\dot{q}_{i=2}(t) = \dot{\varphi}(t)$	okamžitý úhlový kmitočet tyče	[rad . s^{-1}]

[5]

Po zderivování a dosazení do Lagrangeovy rovnice (8) dostáváme následující relace, tedy okamžitou polohu kuličky:

$$\frac{\partial}{\partial x(t)} \{W(t)\} = \left(m_k + \frac{J_k}{R_k^{\,2}} \right) \ddot{x}(t) - m_k x(t) \dot{\varphi}^2(t) + m_k g \sin(\varphi) \tag{9}$$

[5]

Pro okamžitý úhel sklonu tyče je:

$$\frac{\partial}{\partial\varphi(t)}\{W(t)\}=\left(J_t+m_k x^2(t)\right)\ddot{\varphi}(t)+m_k g x(t)\cos(\varphi) \tag{10}$$

při lagrangiánu:

$$L=E_k-E_k=\frac{1}{2}\dot{x}^2(t)\left(m_k+\frac{J_k}{R_k^{\,2}}\right)+\frac{1}{2}\dot{\varphi}^2(t)\cdot\left(J_t+m_k x^2(t)\right)-k_k g x(t)\sin(\varphi) \tag{11}$$

[5]

Položením rovností mezi relace poloh kuličky (6) a (9) a relace úhlu sklonu tyče (7) a (10), lze získat následující relace:

$$-b\dot{x}(t)-\frac{\xi}{R_k}F_n sign(\dot{x}(t))=\left(m_k+\frac{J_k}{R_k^{\,2}}\right)\ddot{x}(t)-m_k x(t)\dot{\varphi}^2(t)+m_k g\sin(\varphi) \tag{12}$$

$$F_s l\cos\varphi=\left(J_t+m_k x^2(t)\right)\ddot{\varphi}(t)+m_k g x(t)\cos(\varphi) \tag{13}$$

[5]

Relace (12) vyjadřuje vliv náklonu na pohyb kuličky, relace (13) pak vyjadřuje vliv kuličky na náklon tyče, přičemž tuto relaci zanedbáváme. Pro odvození přenosové funkce této soustavy (vztah mezi náklonem tyče a pozicí kuličky) budeme dále uvažovat relaci (12), kterou lze zredukovat zanedbáním ramena valivého odporu a odstředivé síly kuličky a linearizovat okolo následujícího ekvilibria:

$$\varphi(t)=\dot{\varphi}(t)=x(t)=\dot{x}(t)=0 \tag{14}$$

[5]

Při linearizaci dále uvažujeme:

$$\sin\varphi\cong\varphi \qquad \varphi\in\left\langle 0;\frac{\pi}{36}\right\rangle[rad] \tag{15}$$

Při uvažování relací (14) a (15) lze relaci (12) přepsat do tvaru homogenní OLDR 2. řádu:

$$\left(m_k + \frac{J_k}{R_k^{\,2}} \right) \ddot{x}(t) + b\dot{x}(t) + m_k g \varphi = 0 \tag{16}$$

Přenosová funkce (vnější popis) v Laplaceově transformaci při nulových počátečních podmínkách a volbou nulového odporu prostředí je dána pak dána relací:

$$\hat{G}_s(s) = \frac{\hat{L}\{x(t)\}}{\hat{L}\{\varphi(t)\}} = \frac{\hat{X}(s)}{\hat{\phi}(s)} = \frac{-m_k g}{\left(m_k + \frac{J_k}{R_k^{\,2}} \right) s^2 + bs} = \frac{-F_g}{\left(m_k + \frac{J_k}{R_k^{\,2}} \right) s^2 + bs} =$$

$$= \frac{b_0}{a_2 s^2 + a_1 s} = \frac{-0{,}55}{0{,}077 s^2} = \frac{-7{,}1428}{s^2} \tag{17}$$

Dle očekávání jsme získali astatický systém 2. řádu. Vzhledem k původním rovnicím (12) a (13) lze očekávat nelineární chování kuličky na krajích nakláněné tyče, kde výrazněji působí zanedbaná odstředivá síla. Další významnou nelinearitou je samozřejmě konečná délka tyče a v praxi se projeví i nerovnosti vodících kolejniček.
[5]

3.4 Výběr operačního sytému pro řízení

Aby systém řízení dobře pracoval v rámci NCS, je zapotřebí vhodně zvolit operační systém, na němž bude probíhat komunikace mezi serverem a klientem a též samotné řízení, tedy regulační algoritmus. Takový systém musí splňovat náležité parametry pro vykonávání Real-time úloh a to zejména v našem případě z hlediska přesného časování regulační smyčky. Vhodným kandidátem je operační systém na GNU/Linuxu, společně s nainstalovanými doplňky nesoucí názvy RTAI nebo Xenomai. Tyto na sobě nezávislé balíčky zajišťují, že OS bude co do latence a přesnosti časování spadat do oblasti RTOS, navíc systém udělají plně preemptivní.

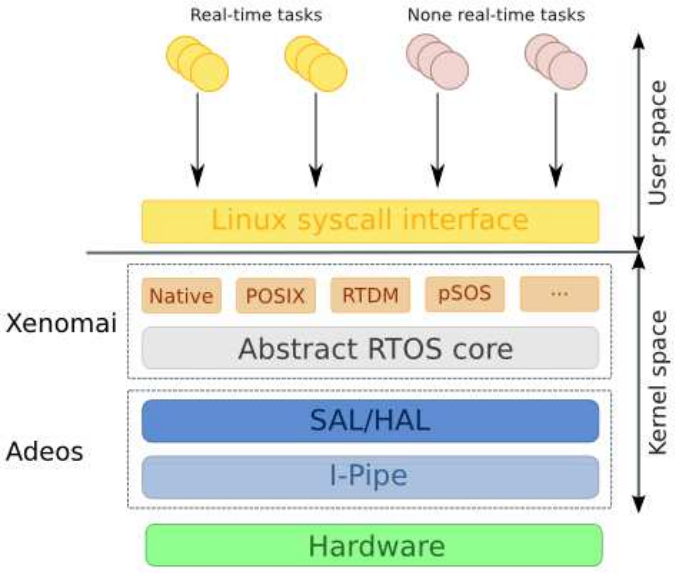
K zajištění správnosti časování je nutné provést v jádře několik změn, např. v zacházení s přerušeními nebo s metodami plánování. Takto můžeme získat Real-time platformu s nízkou prodlevou a plnící náročné požadavky na předvídatelnost v prostředí Linuxu (přístup k TCP/IP, grafické zobrazení a okenní systémy, souborové a databázové systémy atd.).
[9]

Další věc, která se musí na těchto platformách řešit při návrhu řízení založeném na NCS je komunikace. Jak již bylo v předchozí kapitole uvedeno, klasické přístupy komunikace protokolem TCP/IP nejsou vhodné v aplikacích Real-time, kde je nutností, aby byl čas nad strávenými funkcemi vždy stejný, případné možnosti využití časových známek nebo definování QoS. Jeden z takových protokolů obsahuje doplněk pro Linux s názvem RTnet, na jehož vhodnost a kompatibilitu se též odkazuji RTAI i Xenomai.

3.4.1 Xenomai

Je Real-time framework pro vývoj, spolupracující s Linuxovým jádrem, s cílem poskytnout pronikavou, Hard Real-time podporu aplikacím běžících v uživatelském prostoru, integrovanou do systému GNU / prostředí Linuxu. Xenomai je založen na abstraktním RTOS jádře, použitelné pro budování jakékoliv Real-time rozhraní, přes jádro. Ono abstraktní jádro potom vyvá soubor všeobecných služeb RTOS. Libovolný počet RTOS osobností tzv. "skinů" pak může být postavena na jádru, které poskytuje své vlastní specifické rozhraní k aplikacím, které využívají služeb jednoho obecného jádra k jeho provedení.

Xenomai je v podstatě přídavný modul, který využívá služeb HAL, což je abstraktní vrstva nad hardwarem. Snaží se o přesunutí běhu procesů z prostoru jádra do uživatelského prostoru, a taky dokáže tyto ne-Linuxové systémy emulovat. Tím lze zjednodušit přechod aplikací z jiných RTOS na systémy založené na Linuxu. [8], [9]



Obr. 3.4.2 Architektura Xenomai [22]

Xenomai je, jak už bylo naznačeno v předcházející kapitole, je pokračující vývojová větev projektu RTAI, která se zaměřuje na vyplnění mezery mezi tradičním přístupem přes jádro pro dosažení omezeného jittru v nejhorších případech a dlouholeté úsilí o snížení latence původního (vanilla) jádra Linuxu.
[11]

Prvním cílem Xenomai je odsunout aplikace vyžadující Real-time záruky z prostoru jádra (kernel-space), kde byly při starém způsobu přístupu s využitím pomocného jádra uzamčeny. Namísto toho jim umožňuje využít běžný linuxový programovací model v uživatelském prostoru (user-space), zatímco bude stále garantováno omezení nejhoršího zpoždění. Xenomai se zaměřuje na doplnění podpory low latency pro velmi náročné Real-time aplikace, které vyžadují, aby maximální jitter zůstal bez výjimky v rozsahu pár desetin mikrosekund za jakýchkoliv okolností a musí být přitom provozuschopné v uživatelském prostoru (user-space).
[11]

Druhý hlavní cíl Xenomai je zjednodušení přechodu aplikací běžících v tradičních RTOS (jako např. VxWorks, pSOS+, VRTX a podobně) na systémy založené na Linuxu. To je možné díky poskytnutí efektivních emulací API nad RTOS abstrakční vrstvou a dodržovaní standardu POSIX, což umožňuje přenos rozhraní pro operační systémy, standardizované jako IEEE 1003 a ISO/IEC 9945. Vychází ze systémů UNIX, a určuje, jak mají POSIX - konformní systémy vypadat, co mají umět, co se jak dělá apod. Jelikož je velká většina těchto systémů implementována podobným způsobem, je možné vymodelovat obecný set základních vlastností a později je podle libosti upravit pro napodobení různých API.
[11]

3.4.2 RTAI vs Xenomai

Projekt Xenomai byl zahájen v srpnu 2001. V roce 2003 se spojil s projektem RTAI se záměrem vytvořit volně šiřitelnou průmyslovou Real-time platformu pro GNU/Linux s názvem RTAI/fusion na bázi abstraktního Xenomai RTOS jádra. Časem projekt RTAI/fusion usiloval o nezávislost vůči RTAI a v roce 2005 znovu nese jméno Xenomai.
[9]

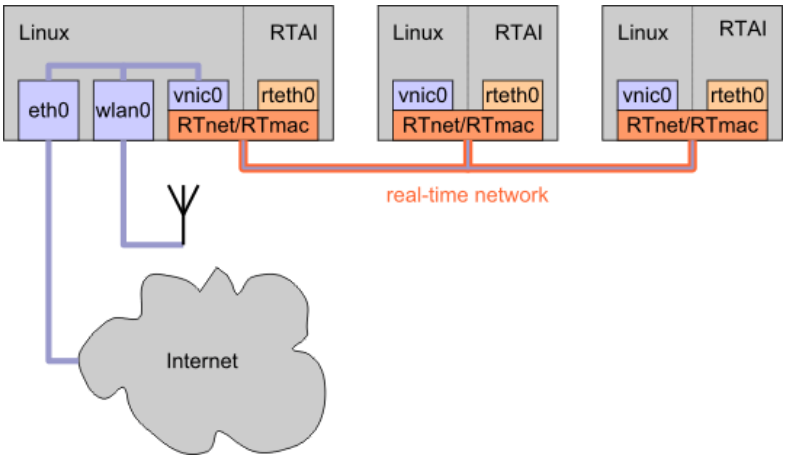
Původním záměrem v projektu bylo nainstalovat a využívat k řízení RTAI, ovšem po několika neúspěšných instalacích, souvisících se vzájemnou nekompatibilitou zastaralého RTAI projektu a již novými Linuxovými jádry byl bez větších problémů úspěšně nainstalován Xenomai ver. 2.5.4.

3.4.3 RTnet

RTnet je open source Hard Real-time síťový protokol pro systémy Linux s nástavbou Xenomai a RTAI. Využívá standardního hardwaru Ethernet a podporuje několik populárních síťových karet NIC chip, včetně Gigabitového Ethernetu. Navíc, Ethernet-over-1394 je k dispozici na základě RT-stack FireWire protokolu.

RTnet implementuje tyto komunikační protokoly: UDP/IP, TCP/IP (základní funkce), ICMP a ARP deterministickým způsobem, např. pomocí TDMA (time division multiple access). Poskytuje POSIX socket API pro Real-time procesy uživatelského prostoru a moduly jádra. Aby se předešlo nepředvídatelným kolizím na síti Ethernet, dodatková vrstva zvaná RTmac kontroluje přístupy k médiím. Pro garanci určitých zpoždění přenosu je nutné využít oddělený komunikační segment.
[8]

Obr.3.4.5a ukazuje typickou konfiguraci RTnet sítě v reálném čase. K dosažení deterministické chování, je zapotřebí speciální síťový segment. Non-Real-time komunikace bude tunelována prostřednictvím tohoto segmentu. Jeden ze tří stanic v reálném čase je nakonfigurována jako vstupní brána do libovolné non-Real-time sítě.

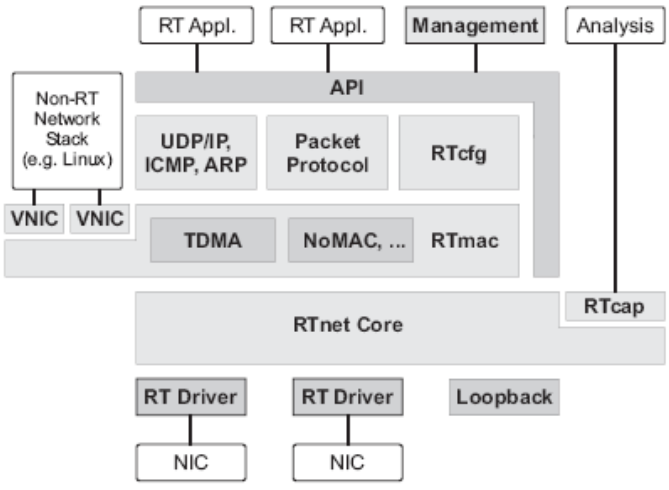


Obr.3.4.5a Typická konfigurace sítě s RTnet v reálném čase

Projekt RTnet byl založen na univerzitě v Hannoveru ze dvou důvodů. Aby bylo možné vyplnit mezeru mezi silně deterministickým hardwarovým řešení a čistě softwarové bázi procesů pro měkké Real-time požadavky.

RTnet poskytuje rámec, který se vyznačuje protokoly a nástroji pro konfiguraci, provozu a diagnostiku v reálném čase Ethernetových sítí. Nabízí „tvrdý“ Real-time přes Ethernet nemodifikovaných složek, protože se nachází v nejmenších vestavěných kontrolérech, v průmyslových počítačích nebo dokonce ve vhodných kancelářských počítačích. Na požádání mohou být účastníci připojeni do switchů nebo hubů standardní topologie „Stars“ a „Bus“.

V běžných sítích Ethernet nastane nedeterministické chování po srážce. Když se dvě stanice pokusí o odeslání současně, nový pokus o odeslání se provádí po náhodné čekací době. Na jedné straně RTnet dosáhne shody v reálném čase díky přesné kontrole vysílajícího okamžiku zpráv. Proto nemůže být kolizím na síti a přetížení jednotlivých účastníků nebo části infrastruktury zabráněno výhradně preventivním vyloučením použití standardních switchů. Modulárně integrovaný přístup k procesu rozhodne, jaký protokol přístupu k médiím bude následovat. RTnet nabízí synchronizační techniku TDMA s probíhající verzi, která přiřazuje pevný časový úsek pro každého účastníka. Tito účastníci jsou synchronizováni Masterem, který signalizuje začátek nového cyklu periodicky se startem Frame zprávy. Spolu se začátkem Frame balíčku, jsou časy účastníků synchronizovány. Podle jejich priority se pack-to-send seřadí podle zúčastněných uzlů. Nejnižší úroveň priority mají rozmanitá a časově nekritická data, jako jsou TCP/IP balíčky, to aby byly předány pouze tehdy, když jsou mimo časově kritickou komunikaci.



Obr.3.4.5b Architektura komponent RTnet

Na druhou stranu, RTnet dosahuje tvrdých Real-time vlastnosti se silně deterministickým plněním zásobníku protokolu. Konvenční protokolové zásobníky vykazují značně konkurenční použitých zdrojů, jako jsou datové vyrovnávací paměti, CPU čas a logické adresy nebo příchozí

fronty. V zájmu zajištění vnucených reaktivit mezi bloky, byly kritické komponenty důkladně optimalizovány, pokud jde o maximální dobu provedení. RTnet tak nabízí vyhovující realizaci v reálném čase UDP / IP, ICMP a ARP. Tímto způsobem je zajištěno, že v každém případě je balíček fyzicky umístěn na kabelu v deterministický čas při odesílání z aplikace. Zvláštní nároky jsou představovány spuštěním kompatibilní sítě v reálném čase s cyklickým datovým provozem. Konfigurace účastníků musí být rozdělena stejně jako jejich funkce dohlížení. RTnet si to uvědomuje pomocí tzv. RTcfg protokolu. To umožňuje centrální projev všech síťových parametrů a jejich distribuce k nově připojeným účastníků bez porušení Real-time požadavků. Navíc se specifické nastavení RTnetu dá společně uživatelsky definovat údai, jako provozní parametry nebo se dají programy převést. RTcfg byl vyvinut jako samostatné složky v rámci RTnet a je nezávislý na použitém transportní protokolu nebo Media Access disciplíně. Zejména podporuje přístup procesů k výměně poškozených nebo nenakonfigurovaných účastníků bez přerušení v běhu provozu v reálném čase.
[15], [16]

3.4.4 Linux verze jádra 2.6.32

Linux je svobodný operační systém unixového typu, jehož původní jádro OS bylo naprogramováno Linusem Torvaldsem v roce 1991 a nadále je rozvíjeno vývojářem z celého světa. V dnešní době jsou podporovány nejrůznější typy platforem.

Všemu ale předcházelo založení projektu GNU, jehož cílem bylo vytvořit nový operační systém unixového typu, který by byl složen jen ze svobodného software. Otcem a zakladatelem projektu je Richard Matthew Stallman. Za tímto účelem sepsal Stallman novou licenci GNU GPL (General Public License)., pod kterou jsou volně šiřitelné všechny části systému GNU. Plnohodnotný operační systém tak vznikl spojení systému GNU s linuxovým jádrem, proto je správné označení GNU/Linux.
[12], [13]

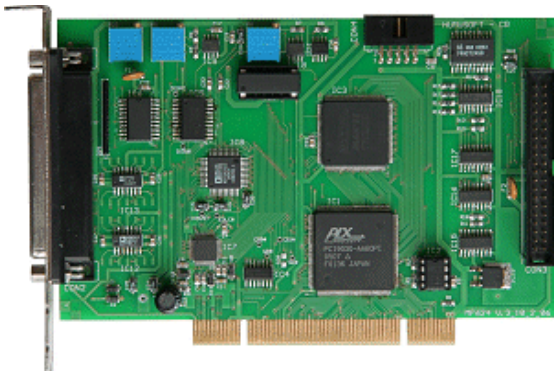
Vývoj řady Linoxového jádra 2.6.32 je sice oficiálně ukončen, ale ještě úplně nekončí, jen přešel do režimu „rozšířené podpory“. Správu převzal Willy Tarreau a chce pokračovat ve vydávání, ale už ne tak často jako jeho předchůdce. Ve zkratce to znamená, že kritické opravy budou vydány rychle. Tato řada se objevila v prosinci 2009 a podporována byla přes dva roky.
[14]

Sudá čísla v označení na druhé pozici znamenají stabilní verzi. I když už vyšlo jádro ve verzi 3.x.xx, stabilita, rozšířená podpora odzkoušeného jádra, které je podporováno patchem Xenomai, plně postačuje k tomu, aby dalo za vznik platformě pro spolehlivý řídicí systém.

3.4.5 DAQ MF624

HUMUSOFT s.r.o. se specializuje na výrobu měřicích karet pro PC, použitelných jak v laboratorních, tak v průmyslových podmínkách. Všechny tyto karty jsou podporovány produkty Real-Time Windows Target, xPC Target a Real Time Toolbox pro MATLAB. Spolu s těmito programovými balíky tvoří integrované a snadno použitelné prostředí pro vývoj aplikací pro řízení a simulaci v reálném čase.
[10]

- Osm single-ended 14-bitových analogových vstupů
- Osm 14-bitových analogových výstupů
- 8 digitálních vstupů, 8 digitálních výstupů
- 4 vstupy inkrementálních snímačů (diferenciální)
- 4 čítače/časovače
- Nízká spotřeba



Obr. 3.4.5 multifunkční I/O karta MF624

- Ovladač pro Real Time Toolbox pro MATLAB
- Ovladač pro Real-Time Windows Target
- Ovladač pro xPC Target
- Ovladač pro Windows, 32 i 64-bitové aplikace
- Nyní již existují ovladače pro Linux 32, 64-bitové

4 Rešerše možných řešení z hlediska operačního systému

Můžeme říci, že naše úloha nespadá do nijak náročných aplikací z hlediska použití nástrojů, které jsou pro RT úlohy typické (použití mutexů, semaforů, zámků kritických sekcí, synchronizace mezi vlákny), avšak správné časování je nutnost, kterou regulační úlohy vyžaduji. Použití systému s RT vlastnostmi je tedy nevyhnutelné. Pro danou aplikaci tedy postačí běžné PC s DAQ kartou pro vytváření akčního zásahu a vhodně zvoleným operačním systémem reálného času. Více o výběru platformy řídicího systému v kapitole 2.4.
[26]

Časově kritické systémy se vyznačují především tím, že je u nich spolu s funkčně správnou odezvou na daný podnět požadováno také *včasné* poskytnutí této odezvy. Běžně jsou označovány jako *systémy pracující v reálném čase*, *systémy pracující s reálným časem* či *systémy pro řízení v reálném čase* – zkráceně také *systémy reálného času*, popř. *systémy RT*, v zahraniční literatuře *RTS* (*Real Time Systems*).
[26]

4.1 Parametry Real-time systémů

Abychom mohli konstatovat, že daný systém pracuje v RT oblasti a je tedy v našem případě vhodný i pro běh regulačních algoritmů, musí za každých podmínek dodržet specifické časy, které souvisí s charakterem úloh na něm spouštěných jak asynchronně tak periodicky. Definují se tedy pojmy:

- Release time, což je minimální prodleva za kterou se RT úloha začne konat.
- Deadline, což je čas, ve kterém musí být úloha dokončena.
- Latence, což je doba mezi příchodem požadavku a jeho provedením, například z hlediska události.
- jitter, což je nechtěná odchylka od stanovené periody.

RT systémy se ještě dají rozdělit na Hard a Soft Real-time. Pokud je na systém kladen požadavek vždy dodržet Release time a Deadline, jde o Hard Real-time systém. Když však chceme dodržet „pouze“ jakési stanovené průměrné hodnoty výše zmíněných parametrů v jednom případě nebo v druhém, dodržet maximální počet nedodržení Dealine za jakýsi časový úsek, potom mluvíme o Soft Real-time systémech.

4.2 Příklady a koncepce různých Real-time OS

4.2.1 PikeOS

PikeOS je operační systém pracující v reálném čase pro bezpečnostně kritické aplikace speciálně vyvinutý pro snadné oddělení či verifikaci jednotlivých částí elektronického systému. Je založený na mikrojádře a používá se převážně v embedded systémech a serverech.

PikeOS zajišťuje, aby byly všechny dostupné prostředky, jako například operační paměť, jednoznačně rozděleny. Jednotlivé přihrádky (partitions) mají přístup výhradně na tyto přidělené systémové zdroje. Přihrádky si však mohou, je-li to požadováno, vyměňovat data či poskytovat zdroje přes zabezpečené konfigurovatelné komunikační kanály. Deterministické chování a konfigurovatelné paravirtualizační schopnosti PikeOS umožňují široké využití v letectví a dopravní a automobilové technice (MPC5200), kosmonautice (LEON3), zdravotnictví (X86), automatizaci (ARM) a dalších odvětvích.
[17], [18]

4.2.2 Windows CE

Windows CE je preemptivní více-úlohový systém, který byl navržen jako operační systém pro obecné použití. Prvotně je určen pro jednoduchá zařízení neobsahující rotující paměťová média. Vzhledem k tomu, že nebyl vytvořen pevně pro jeden typ hardwaru, musí vždy mezi operačním systémem a vlastním zařízením existovat speciální unifikující vrstva softwaru, která je nazývána OEM adaptivnívrstva .

Microsoft Windows CE prošel velmi významným vývojem, zejména co se týče rozdílů mezi verzemi 2.12 a 3.0 v oblasti RT. Jeho posun ke skutečnému RTOS je velmi citelný. Windows 2.12 lze hodnotit jako operační systém splňující minimální požadavky kladené na RTOS a může být zcela jistě použit v určitých typech úloh vyžadujících RT vlastnosti. Splňuje především tyto požadavky:

- je multithreadový a preemptivní
- podporuje osm úrovní priorit vláken
- podporuje dědičnost priorit, která umožňuje dynamickou změnu pro realizaci inverze priorit
- podporuje synchronizační mechanismy s deterministickým chováním
- podporuje programovatelné časovače a přístup k vysoce výkonným časovačům
- umožňuje dvouúrovňové zpracování přerušení pro rychlé odezvy
- latence přerušení a jejich zpracování jsou determinovatelné a pevné

- délka volání jádra je deterministická, pevná a nezávislá na počtu aktivních systémových objektů

I přes všechny tyto pozitivní vlastnosti má verze 2.12 několik nedostatků, které někteří odborníci považují za velmi závažné. Jsou jimi zejména:

- počet úrovní priorit vláken je příliš nízký
- přerušení nemohou být vnořena
- latence přerušení je příliš vysoká

S uvedenými výtkami nelze než souhlasit, avšak je nutné dodat, že nikde nejsou exaktně a absolutně stanoveny hodnoty, kterých by tyto parametry měly nabývat (zejména co se týče počtů priorit či latence). Pro typy úloh, pro něž je daný počet priorit dostatečný a časové odezvy přiměřené, je daná verze systému naprosto dostačující.

Ve verzi 3.0 však nastaly výrazné změny, a to zejména v uvedených třech sporných parametrech. Jak jsme si ukázali, tyto parametry nabývají těchto hodnot:

- počet úrovní priorit je zvýšen na 255
- přerušení mohou být vnořená
- latence přerušení byla výrazně snížena

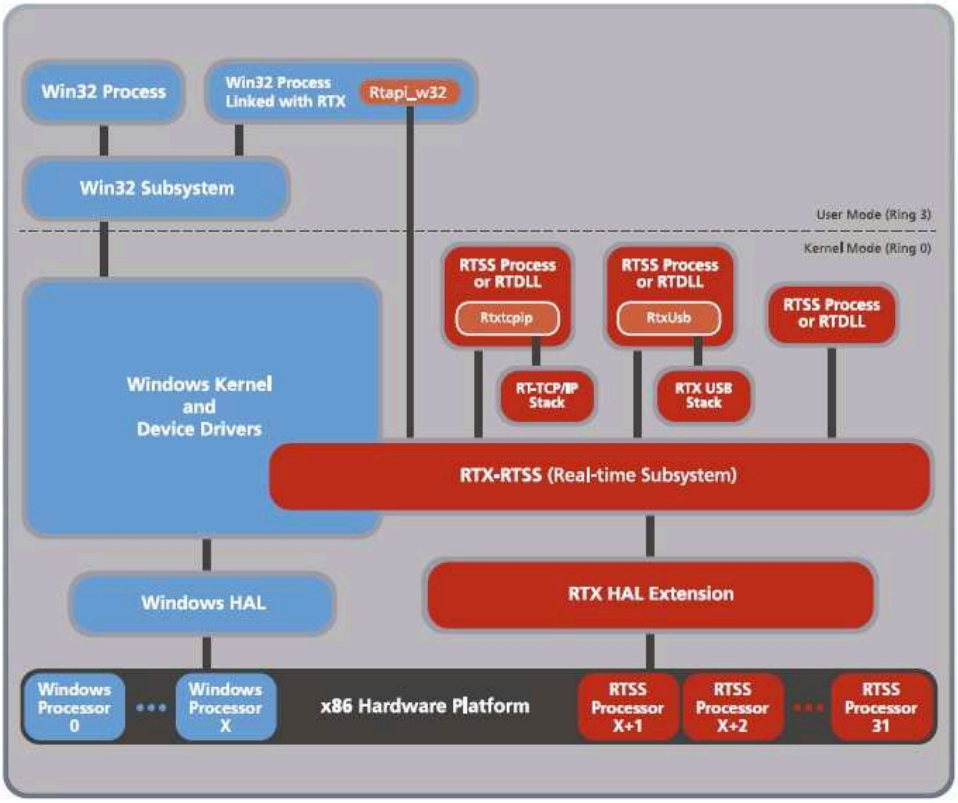
[19]

4.2.3 Windows s použitím RTX

Operační systém Windows NT/2000/XP od společnosti Microsoft je z mnoha důvodů stále častěji považován za platformu dalšího rozvoje systémů pro řízení v reálném čase. Z důvodu přísných požadavků na reakční dobu systémů reálného času je ale nezbytné možnosti Windows NT/2000 v tomto směru dále rozšířit. Podívejme se, jak lze k implementaci subsystému reálného času do prostředí Windows NT/2000 použít doplněk RTX od americké společnosti Ventur. RTX implementuje deterministické plánování vláken reálného času, komunikační mechanismy mezi prostředím reálného času a přirozeným prostředím NT/2000 a další rozšíření Windows NT/2000 o vlastnosti, které lze často najít ve specializovaných operačních systémech reálného času.

Stručně řečeno, rozšíření RTX je implementováno jako kolekce knihoven (statických i dynamických) a pomocí subsystému reálného času (RTSS) jako ovládač zařízení jádra NT a rozšířená HAL (obr. 4.2.3). Subsystém RTSS implementuje jak objekty reálného času, tak již dříve zmíněný plánovač. Knihovny poskytují přístup k subsystému prostřednictvím rozhraní RTX API, které tak uvedené objekty zpřístupňuje. Všimněme si, že RTX API lze vyvolat i ze standardního prostředí Win32 stejně jako v rámci RTSS. Při používání RTX API z Win32

se neposkytuje determinismus dostupný v rámci RTSS. Větší objem aplikačního vývoje lze spíše než v prostředí poskytovaném DDK realizovat v komfortnějším programovacím prostředí Win32. Vše, co je nutné ke konverzi programu používajícího Win32 do programu používajícího RTSS, je znovu aplikaci spojit (link), tentokrát s jinou kolekcí knihoven.



Obr. 4.2.3 Architektura Windows RTX na platformě x86

Aplikace RTX – stejně tak jako RTX samo o sobě – jsou implementovány na vrchol zaveditelných ovládačů NT, ačkoliv vyžadují propojení (hooks) s manažerem I/O. Toto je přirozený stav: ovládače Windows NT jsou z pohledu manažeru služeb NT uživatelsky nastavitelné a lze je zavést do adresového prostoru jádra.
[20]

4.2.4 RTLinux

RTLinux je malý a rychlý operační systém, který je v souladu s normou POSIX 1003.13, což je architektura pro minimální operační systémy reálného času. RTLinux lze považovat za úplný operační systém s předvídatelnou činností v reálném čase, bez rozhraní pro standardní Linux bez reálného času. Vlákna RTLinuxu jsou zpracovány přímo plánovacím algoritmem s pevnou prioritou. Jádro a všechny procesy standardního Linuxu jsou řízeny plánovačem RTLinuxu jako úlohy v pozadí. RTLinux vytváří úplný obecný operační systém, který běží nad malým předem definovaným jádrem RTOS.

RTLinux druhé verze je strukturován jako soubor volitelných komponentů doplňujících jádro, které povoluje instalaci obslužných rutin přerušení s velmi nízkou prodlevou (Low Latency). Tento hlavní komponent byl rozšířen o podporu SMP a zároveň je zjednodušen odebráním některých vlastností, které mohou být poskytovány jiným způsobem. Hlavní funkčnost RTLinuxu spočívá v kolekci modulů poskytujících volitelné služby a úrovně abstrakce.

Tyto moduly zahrnují:

- rtl sched - plánovač priorit, který podporuje "jako POSIXové" rozhraní a původní v1 RTLinux API.
- rtl time - který řídí hodinový signál procesoru a exportuje abstraktní rozhraní pro připojení obslužných rutin na hodinový signál.
- rtl posixio - podporuje POSIX styl read/write/open rozhraní pro ovladače zařízení.
- rtl fifo - propojuje RT úlohy a obslužné rutiny přerušení k linuxovým procesům přes vrstvu zařízení, takže linuxové procesy mohou číst a zapisovat do RT komponent.
- semaphore - dává RT úlohám blokovací semafory (synchronizační objekty, které slouží k synchronizaci přístupu ke sdíleným prostředkům). Podpora POSIXových mutexů bude taky k dispozici.
- mbuff - poskytuje sdílenou paměť mezi RT komponenty a linuxovými procesy (Yodaiken, V., Barabanov, M.).

Klíčovým cílem návrhu RTLinuxu je transparentní, modulární a rozšiřitelný systém. Transparentnost znamená, že zde nejsou žádné neotevíratelné černé skříňky a cena každé operace by měla být stanovitelná. Modularita znamená, že je možné vynechat nepotřebnou funkci a ušetřit tak výkon. Základní RTLinux systém podporuje pouze vysokorychlostní obsluhu přerušení. Rozšiřitelnost znamená, že programátoři by měli být schopni přidat moduly a upravit si systém podle vlastních požadavků.
[9], [17]

4.2.5 VxWorks 6.x

je operační systém pro řízení v reálném čase, který se vyznačuje RT mikrojádrem wind. Toto jádro zahrnuje většinu nástrojů pro podporu reálného času. Patří k nejrozšířenějším operačním systémům reálného času zejména v oblasti průmyslových aplikací embedded systémů. Mezi základní charakteristiky patří:

- Neomezený počet procesů/úloh
- Preemptivní plánování
- 256 úrovní priorit
- rychlá a flexibilní meziprocesová komunikace
- dědění priorit
- fronty zpráv
- signály
- roury
- sockety
- podporovaná CPU: PowerPC, ARM, Intel x86, atd.
- není POSIX kompatibilní (řeší některé nedostatky specifikace)

Mikrojádro je navrženo s minimální režii systému, což umožňuje rychlou a deterministickou odezvu na externí událost. Tento systém je bezpečný i při použití v kritických aplikačních úlohách (byl použit v aplikacích meziplanetárního výzkumu). Systém je kompatibilní s řadou průmyslových standardů a lze jej používat na běžných CPU

5 Sestavení základního regulačního řetězce

5.1 Návrh PID regulátoru

Pro syntézu regulátoru a jeho simulaci byl zvolen program MATLAB / Simulink. Funkce *pidtune, pidtool,* které jsou součásti verze 2010b. Pomocí těchto funkcí jsme schopni dojít ke konstantám regulátoru v paralelním tvaru $G_R(s) = K_P + \frac{K_I}{s} + K_D \cdot s$.

Nutností je, aby regulátor obsahoval derivační složku, protože model soustavy je astatický systém druhého řádu a má tedy přirozený integrační charakter. Aby měl průběh na výstupu soustavy v uzavřené smyčce uspokojivý průběh, na výstupu soustavy by se měl objevit co možná nejmenší překmit, aby bylo možné řídit pohyb kuličky na tyčí v co největším rozsahu. Celková doba ustálení pak bude druhořadá a hlavně bude omezená saturací pohybu ramene. Úkol minimalizace času uspokojivého ustálení na výstupu soustavy však zůstane.

Vstupními parametry funkce pro získání PID konstant specifikujeme frekvenci překmitu, fázovou bezpečnost a typ regulátoru. Parametry byly zvoleny takto:

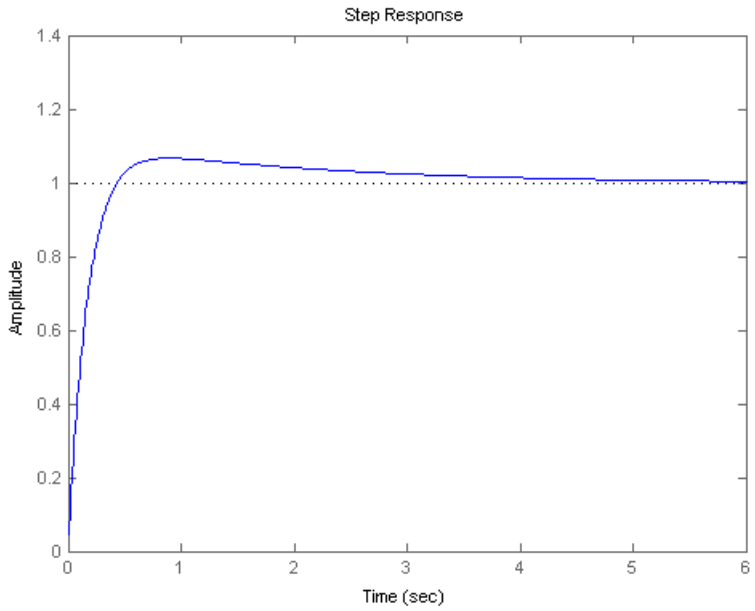
Typ regulátoru: *PID*
Crossover frequency = *6.35 rad/s*
Phase Margin = *85°*

Kód Matlabu je součástí přílohy 1:

Výstup z Matlabu:

```
Continuous-time PID controller in parallel form:

                 1
Kp + Ki* --- + Kd * s
                 s

with Kp = 0.49201, Ki = 0.043728, Kd = 0.88671


info =

              Stable: 1
    CrossoverFrequency: 6.3500
           PhaseMargin: 85
```

```
Transfer function:
   6.334 s^2 + 3.514 s + 0.3123
-----------------------------------
s^3 + 6.334 s^2 + 3.514 s + 0.3123
```



Obr. 5.1 Odezva uzavřené regulační smyčky na jednotkový skok

Tyto konstanty pak v nasazení regulátoru na reálnou soustavu budou jistě upravovány a laděny, protože na matematickém modelu se neprojevuje nelinearita, tření ani saturace dána konstrukcí.

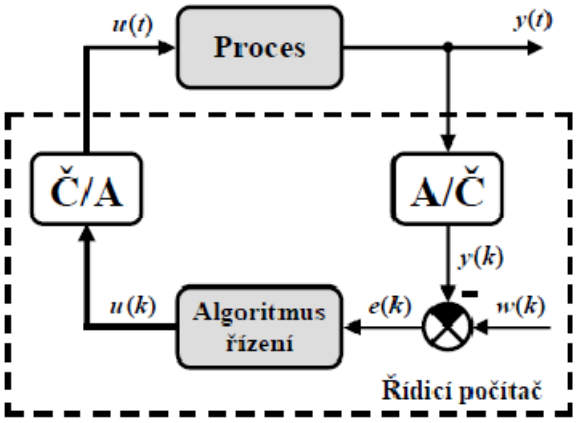
5.2 Číslicové řízení

Protože výsledný řídicí algoritmus se bude vykonávat v aplikaci, která poběží pod operačním systém Linux, je převod algoritmů na číslicové řízení nutnosti. Jedním z důvodů zvolení PSD algoritmu je vývoj programů v jazyce ANSI C, v němž naprogramovat PSD (proporciálně-sumačně-diferenční) algoritmus není příliš složité.
[24]

Číslicový počítač pracuje s informací ve formě čísel, kterou mu poskytuje analogově-číslicový převodník A/Č (viz obr.5.2a), který snímá okamžitou hodnotu spojitého průběhu regulované veličiny *y(t)*. Převodník pracuje na stejném principu jako např. digitální voltmetr, který převádí spojitý průběh měřeného napětí do číselného tvaru. Snímání probíhá obvykle

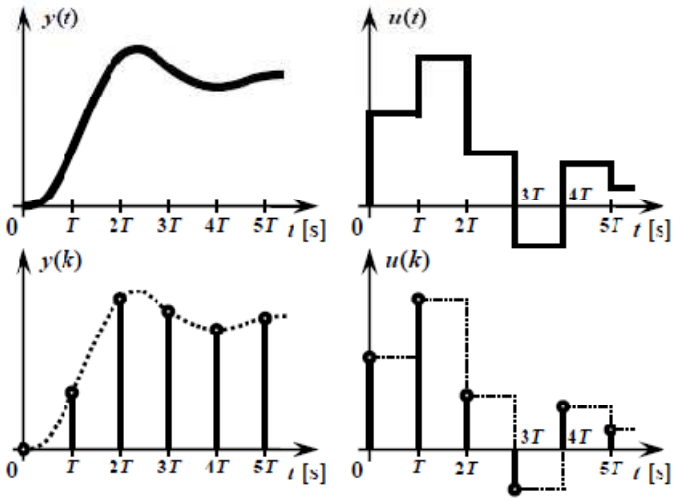
v pravidelných časových okamžicích s tzv. periodou vzorkování T. Řídicí počítač dostává informaci o spojitém průběhu *y(t)* ve formě číselné posloupnosti *y(k).*

Vhodnou hodnotu periody vzorkování získáme jednoduše tak, že dobu praktického ustálení přechodové charakteristiky regulované soustavy (regulovaného technologického procesu) rozdělíme na ca 50 - 100 dílků (není kritické). V případě, že je však regulovaná soustava např. kmitavá s vlastními kmity vyšší frekvence nebo je regulovaná veličina *y(t)* zatížena nežádoucím šumem, volíme frekvenci vzorkování tak, abychom vzorkováním zachytili dostatečnou šíři frekvenčního spektra, podle Shannonova teorému musí být frekvence vzorkování alespoň dvakrát vyšší, než je maximální frekvence, kterou vzorkováním chceme zachytit bez zkreslení.
[24]



Obr.5.2a Blokové schéma regulačního obvodu s řídicím počítačem [24]

Numerická hodnota regulované veličiny *y*(*k*) se porovnává s hodnotou řídicí veličiny *w*(*k*) (také v numerickém tvaru). Regulační odchylka *e*(*k*) je přirozenou mírou nesouladu mezi požadavkem a skutečnou hodnotou *y*(*t*) v aktuálním okamžiku a je postupně po krocích ukládána do registru počítače, ve kterém je uloženo vždy několik starších hodnot *e*(*k*). Algoritmus řízení je realizován vhodným programem, který z aktuální a minulých hodnot *e*(*k*) vypočítává aktuální hodnotu řídicí veličiny *u*(*k*). Ta je pomocí číslicově-analogového převodníku Č/A převedena na akční veličinu *u*(*t*). Pomocí akčního členu je následně akční veličina *u*(*t*) převedena na konkrétní fyzikální veličinu, která působí na reálný technologický proces. Vše, se děje opakovaně v pravidelném taktu daném periodou vzorkování *T* (někdy též hovoříme o periodě nebo kroku řízení). Z logiky věci vyplývá, že akční veličina *u*(*t*) je tzv. schodová (po úsecích konstantní) funkce, mění svoji hodnotu vždy v okamžiku nového výpočtu *u*(*k*) a zůstává konstantní až do dalšího okamžiku vzorkování, viz Obr.5.2b.
[24]



Obr.5.2b Možné průběhy signálů v regulačním obvodu [24]

5.3 Odvození PSD regulátoru z konstant PID regulátoru:

Regulátor PSD (Proporcionálně – Sumačně – Diferenční) je diskrétní (numerickou)variantou spojitého PID regulátoru. Spojitý řídicí algoritmus PID regulátoru

$$u(t) = r_0 e(t) + r_1 \int_0^t e(\tau)d\tau + r_2 \frac{de(t)}{dt}$$

je přibližně nahrazen diskrétním výpočtem v *k*-tém regulačním kroku (v čase $t = kT$). Jestliže:

$$r_2 \frac{de(t)}{dt} \approx r_2 \frac{e(kT) - e[(k-1)T]}{T}$$

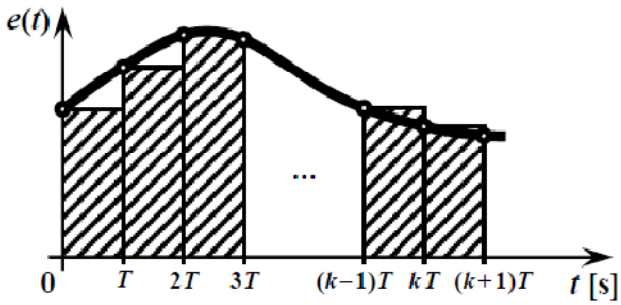
$$r_1 \int_0^t e(\tau)d\tau \approx r_1\{e(0)T + e(T)T + e(2T)T + \cdots + e[(k-1)T]T\} = r_1 \sum_{i=0}^{k-1} e(iT)$$

[24]

Pak je akční veličina v *k*-tem kroku vyjádřená takto:

$$u(kT) = r_0\, e(kT) + r_1 \sum_{i=0}^{k-1} e(iT) + r_2 \frac{e(kT) - e[(k-1)T]}{T}$$

Nevýhodou takovéhoto (tzv. nerekurzivního) řídicího algoritmu je nutnost si pamatovat celou historii vývoje regulační odchylky *e*(*iT*). Velmi elegantně lze tuto nevýhodu obejít výpočtem tzv. akčního zásahu, změny (diference) akční veličiny *u*(*kT*) v aktuálním *k*-tém regulačním kroku.
[24]



Obr. 5.3 Diskrétní náhrada regulátoru [24]

Vypočtení diference:

$$\begin{aligned}
\nabla u(kT) &= u(kT) - u[(k-1)T] \\
&= r_0\, e(kT) + r_1 \sum_{i=0}^{k-1} e(iT) + r_2 \frac{e(kT) - e[(k-1)T]}{T} - \\
&- r_0\, e(kT) - r_1 \sum_{i=0}^{k-2} e(iT) - r_2 \frac{e[(k-1)T] - e[(k-2)T]}{T} = \\
&= e(kT)\left[r_0 + \frac{r_2}{T}\right] + e[(k-1)T]\left[r_1 T - r_0 - \frac{2r_2}{T}\right] + e[(k-2)T]\frac{r_2}{T} = \\
&= b_0 e(kT) + b_1\, e[(k-1)T] + b_2 e[(k-2)T]
\end{aligned}$$

Kde:

$$b_0 = r_0 + \frac{r_2}{T}, \qquad b_1 = r_1 T - r_0 - \frac{2r_2}{T}, \qquad b_2 = \frac{r_2}{T}$$

[24]

Následným přičtením vypočtené diference k minulé hodnotě akční veličiny $u[(k-1)T]$ získáme její aktuální hodnotu:

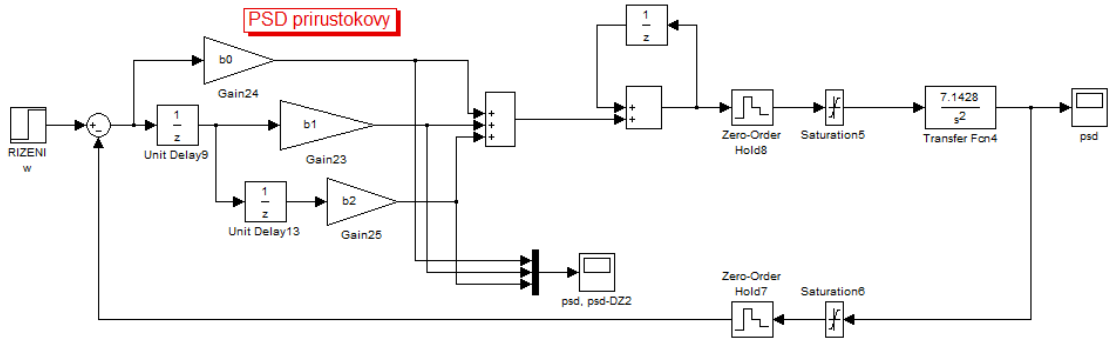
$$u(kT) = u[(k-1)T] + \nabla u(kT)$$

Takto vytvořený rekurzivní algoritmus potřebuje k výpočtu akční veličiny $u(kT)$ pouze současnou a dvě starší hodnoty regulační odchylky $e(kT)$, $e[(k-1)T]$ a $e[(k-2)T]$. Výpočet je velmi rychlý, bez vyšších nároků na paměť počítače. Známe-li seřízení spojitého PID regulátoru, lze velmi snadno nalézt při daném regulačním kroku jeho PSD diskrétní ekvivalent podle uvedených vztahů.
[24]

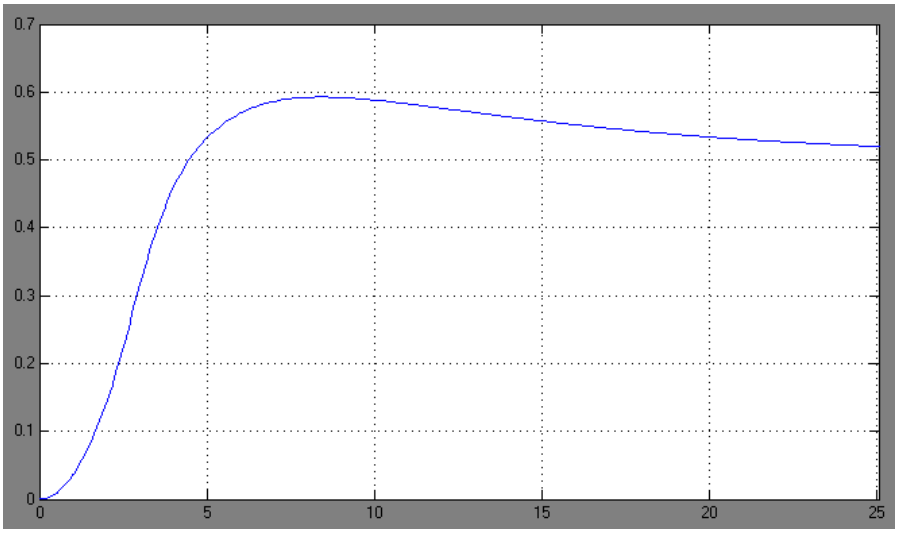
5.3.1 Simulace PSD regulátoru v Matlab/Simulink

Kód v Matlabu, ve kterém dosadíme odvozené konstanty z minulé kapitoly je součásti přílohy 2.

Tento vypočtený regulátoru se pokusíme aplikovat na matematický model v Simulinku, který již obsahuje saturace akční veličiny, kterou v tomto případě je omezení naklonění ramene (tyče), na níž se kulička pohybuje a které činí rozsah úhlu: *u(k)=<-0.01;+0.01> rad*. Další saturace, která se na modelu kulička na tyči nachází, je omezený pohyb kuličky na samotném rameni, které má délku *0.92m*. Ve zpětné vazbě se tedy nachází saturace výstupu *y(k)=<0;0.92> m*.



Obr. 5.3.1a schéma přírůstkového PSD regulátoru v Simulinku:



Obr. 5.3.1b Odezva soustavu s PSD přírůstkovým algoritmem na řídicí veličinu 0.5

Na výstupu soustavy lze pozorovat výrazný Wind-Up efekt, který způsobí sumační složka pří nasycení v saturačním bodě 0,01 nebo -0,01 akční veličiny. Suma $r_1 T \sum_{i=0}^{k-1} e(iT)$ stále načítá svou hodnotu i přes saturaci akční veličiny před soustavou. Při přechodu regulační odchylky do opačných hodnot se začne od této již vysoké sumy odčítat, trvá však několik kroků než se hodnota sumy dostane na saturační mez, tím dojde k jakémusi zpoždění akční veličiny *u(k)* společně s překmitem soustavy.

Proto není přírůstkový algoritmus vhodný pro soustavy s velkým omezením akční veličiny, nelze totiž aplikovat saturaci do celkového přírůstku $\nabla u(kT)$. Sumační větev, ve které má být omezení pro splnění Anti-Wind-Up efektu je v algoritmu schována.

5.3.2 Výpočet PSD s ohledem na Anti-Wind-Up efekt

Jiným řešením, ve kterém již sumační větev lze omezit o danou saturaci, je diskretizace diferenciální rovnice, převedeme tuto rovnici akční veličiny a diferenční integrál nahradíme metodou obdélníků zleva. Perioda vzorkování je jako v minulém případě T. Dostaneme tak přenosovou funkci *F(z).*

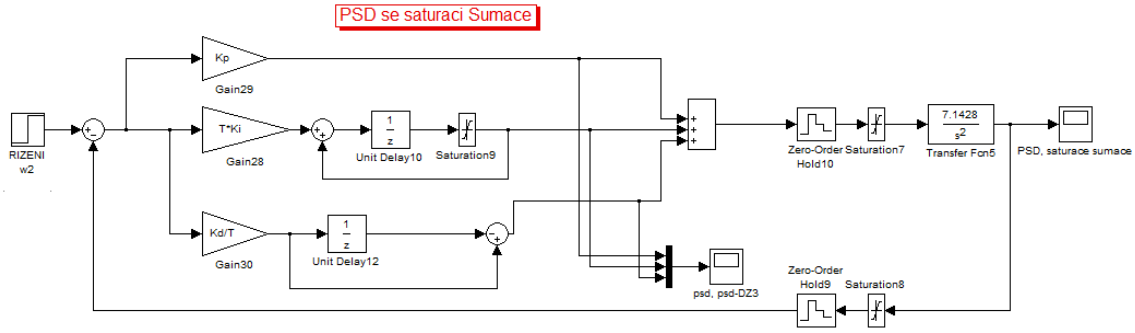
Akční veličina:

$$u(kT) = r_0 e(kT) + r_1 T \sum_{i=0}^{k-1} e(iT) + r_2 \frac{1}{T}\{e(kT) - e[(k-1)]T\}$$

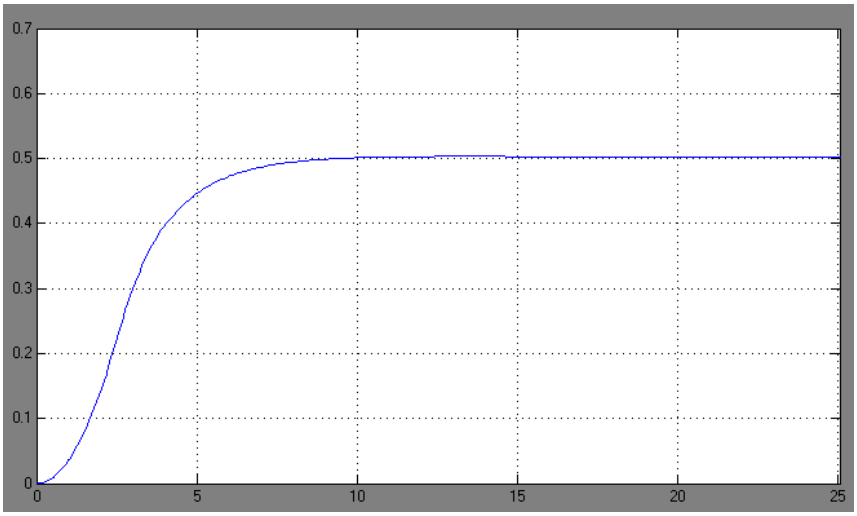
Přenosová funkce:

$$F(z) = r_0 + r_1 T \frac{\mathrm{z}^{-1}}{1 - \mathrm{z}^{-1}} + \frac{r_2}{T}(1 - z^{-1})$$

[6]



Obr. 5.3.2a Schéma PSD regulátoru se saturaci v sumační větvi v Simulinku:



Obr. 5.3.2b Odezva soustavy s regulátorem PSD a Anti-Wind-Up Efektem na řídicí veličinu 0.5

Jde o stejný algoritmus se stejnými konstanty jako v předchozím případě (kapitola 4.4), s tím rozdílem, že můžeme zasáhnout do sumační větve a přidat omezení pro celkovou sumu, která už nebude narůstat nad zadanou mez.

Omezení sumace (saturací) bylo laděním stanoveno tak, aby byl překmit v polovině ramene co nejmenší. Saturace byla stanovena na jednu šestinu úhlu rozsahu ramene <-0.01/6; +0.01/6>.

5.4 Regulační smyčka NCS jako na RT úloha

Úkol jako je vzdálené řízení soustavy s astatismem a možným dopravním zpožděním mezi regulátorem a soustavou, vyžaduje v regulátoru derivační složku pro její integrační charakter. Bez reakce na derivaci $u_d(t) = e(t)/dt$ nebo sumaci $u_s(kT) = 1/T[e(kT) - e[(k-1)T]]$ regulační odchylky, zůstává soustava nestabilní. Zároveň však dopravní zpoždění, které vzniká v našem případě převážně na síti, může mít za následek pozdní akční zásahy, kvůli právě chybně provedené diferenci.

Je zřejmé, že pokud by byla zvolena relativně malá perioda řízení reálného modelu, nemusela by zpráva o požadovaném akčním zásahu přijít k soustavě ve správném kroku. Takové vysoké dopravní zpoždění na síti by nad jistou míru mohlo mít dopad na stabilitu systému.

NCS diskrétní regulaci s dopravním zpožděním jde do jisté míry chápat jako RT úlohu kde by časovou uzávěru (Deadline) představoval časovou vzdálenost mezi dvěma kroky regulace. To znamená, že vystavený kritický čas, do kterého musí být stihnut akční zásah, je čas započetí regulačního kroku o jednu periodu delší. Ten je generován při každé započaté úloze, periodě řízení. Deadline je splněn, pokud je dopravní zpoždění od regulátoru k soustavě menší než je polovina kritické doby, což je v tomto případě perioda řízení, za předpokladu že dopravní zpoždění akční veličiny tj. od regulátoru k soustavě má stejnou hodnotu jako zpoždění od soustavy (zpětná vazba).

Přesněji lze vyjádřit vztahem:

$$(T_{d\,R} + T_{d\,S}) = T_{ping} < (t_k + t_{k+1}) = T_{kritický}\ ,$$

nebo jednoduše: $T_d < \frac{T}{2}$

Kde:

$T_d = T_{d\,R} = T_{d\,S}$

$T = t_k + t_{k+1}$

T_{ping}	čas odezvy v terminologii server klient
$T_{kritický}$	Deadline, kritický čas vykonání úlohy
$T_{d\,R}$	dopravní zpoždění od regulátoru k soustavě
$T_{d\,S}$	dopravní zpoždění od soustavy k regulátoru
t_k	čas v *k*-tém kroku regulace

6 Testy kritických parametrů pro regulaci.

Zjistit přiměřenou periodu řízení T je nutné nejen z důvodu stability řízené soustavy, ale také vzhledem k omezením ze strany operačního systému, který musí danou regulační úlohu spouštět periodicky s co nejmenší chybou. Další omezení NCS plyne z dopravního zpoždění na síti, které musit vždy nižší než zvolená perioda řízení pro deterministické chování soustavy (vysvětleno v kapitole 5.5).

Testy o tom jak rychle se musí řídit vybraný model s vybranými konstanty, aby byla zachována stabilita, byla provedena v simulačním prostředí MatLab/Simulink. Periody, ke kterýmž dojdeme simulací, si ověříme v kapitole 6.2 na reálném modelu také přes prostředí Simulinku, ale již s RT Toolboxem, který obsahuje ovladače k řízení vybrané měřící kartě MF624 od firmy Humusoft.

Jestli pak vybrané platformy Linux+Xenomai a použitá komunikace dovedou garantovat stanovené časy, bude ověřeno vhodnými testy latence, jitteru a pingu, jak při zatíženém operačním systému a sítě tak při systém nezatíženém.

Testy obsahuji pouze části výsledků. Celé výpisy z terminálů (konzolí příkazového řádku), zdrojové kódy, skripty, spustitelné soubory a soubory Matlabu typu *mdl* a *m* lze nalézt v přílohách.

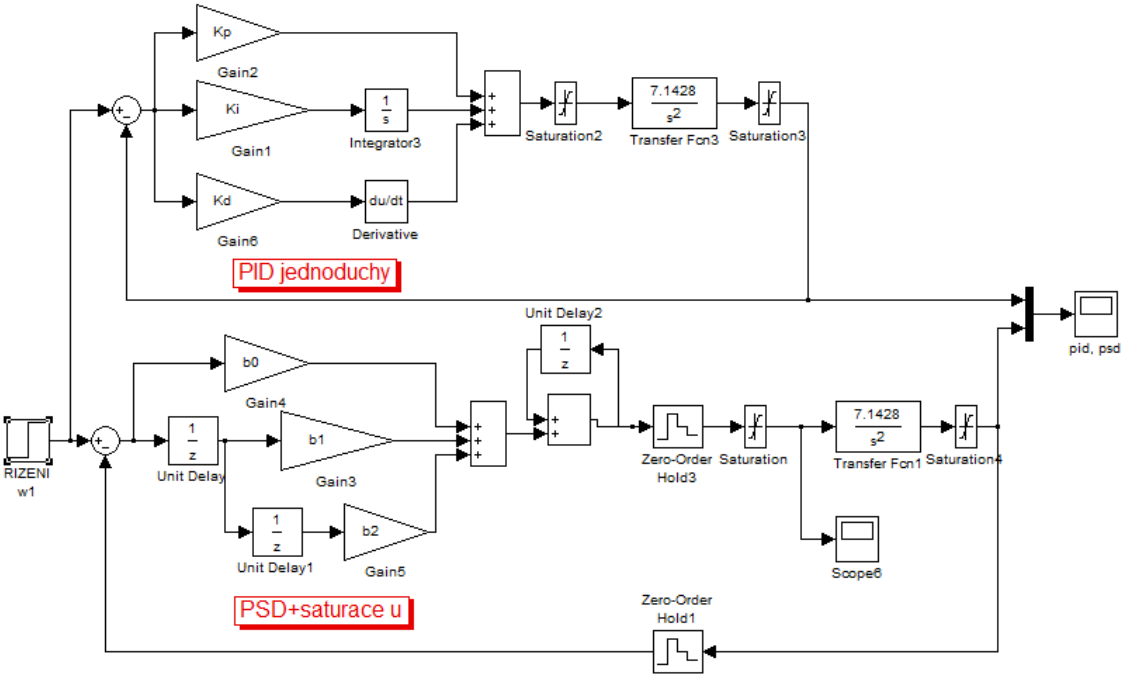
6.1 Testy vedoucí ke stanovení periody řízení pomocí simulace

Protože tyto testy byly prováděny dříve než návrh PID konstant v kapitole 4, následující testy jsou s konstantami odlišnými od minulé kapitoly. Způsob syntézy regulátoru byl však stejný a také prve vedl na přírůstkový PSD regulátor.

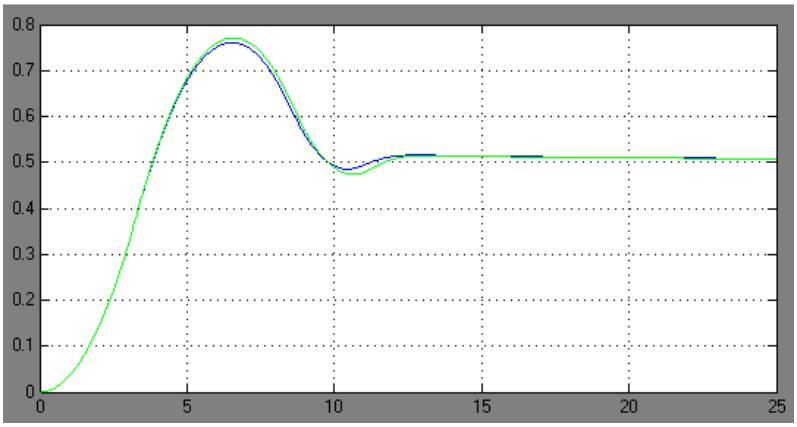
Parametr T (perioda řízení) byl dosazován experimentálně (postupně se snižoval) od hodnoty T=20ms, která již vedla k viditelnému rozdílu na výstupu soustavy oproti uzavřené smyčce PID regulátoru se soustavou ve spojitém čase. Řídicí veličina byla 0.5 (středová poloha) z důvodu omezení výstupu na reálném modelu, na kterém se budou v další kapitole ověřovat stejné periody řízení.

Konstanty regulátoru PID:

$$K_p = 0.49233$$
$$K_i = 0.024776$$
$$K_d = 0.38937$$

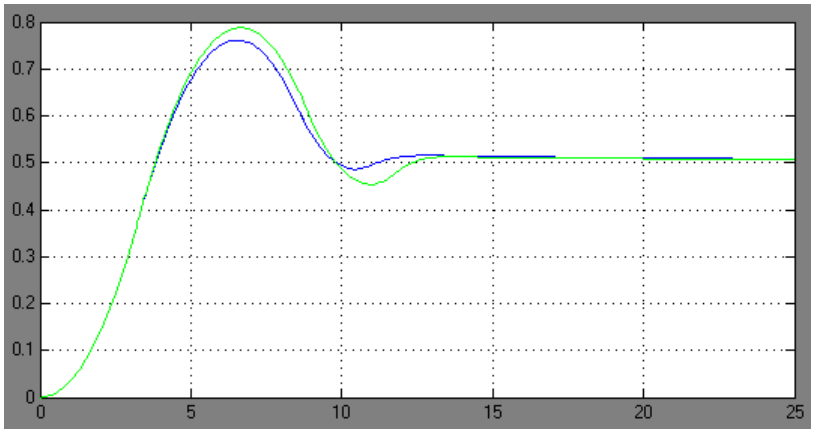


Obr. 6.1a Schéma regulovaných soustav se saturacemi v Simulinku pro stanovení periody řízení



Obr. 6.1b porovnání PID (modře) s PSD regulátorem na výstupu soustavy s T=20ms

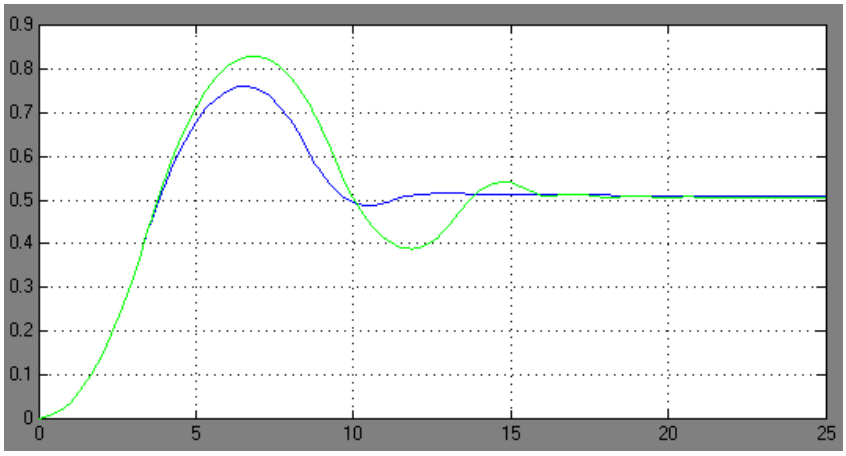
Simulace parametru T = 100ms:



Obr. 6.1c Porovnání PID (modře) s PSD regulátorem na výstupu soustavy s T=100ms

Odezva systému na diskrétní řízení v hodnotě 0,5 s periodou 100ms má již vyšší překmit, soustava však moc nekmitá a doba ustálení je přibližně stejná v porovnání s periodou řízení 20ms. Takové vzdálené řízení by tedy nebylo náročné na transport po síti i bez použití speciálních deterministických protokolů při nezatížené síti. Otázkou je jak se bude chovat reálná soustava s omezeními, jako jsou např. rozdílnost matematického modelu (tření, nerovnost kolejniček, nevelká rychlost servomotoru).

Simulace parametru T=333ms:

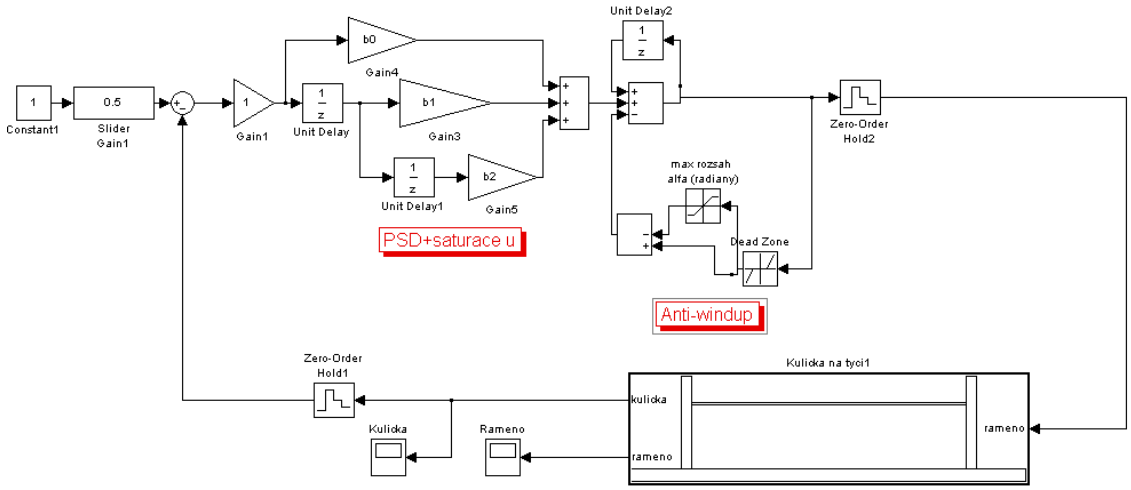


Obr. 6.1d Porovnání PID (modře) s PSD regulátorem na výstupu soustavy s T=333ms

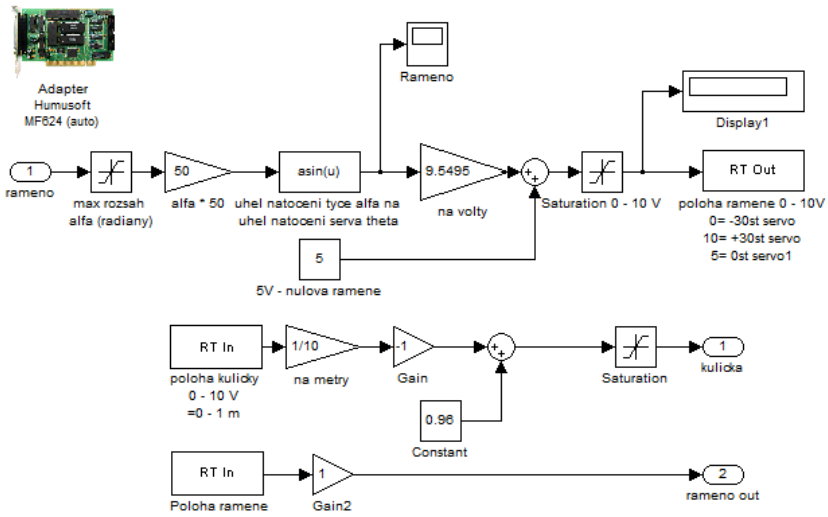
Ukázka ne příliš vhodně zvolené periody řízení (Obr. 6.1d), soustava je sice teoreticky stabilní, ale doba ustálení zde je až kolem 22s, což není pro řízení tohoto modelu dostačující. Teoreticky je soustava s výše uvedeným regulátorem stabilní i při periodě 380ms, v tomto případě se hodnota výstupní veličiny ustálí až někdy po 50s.

6.2 Řízení modelu v RT Toolbox Matlab/Simulink

V minulé kapitole bylo nasimulováno několik řízení modelu s periodami T=20ms, 100ms, 333ms. Ty budou použity pro ověření funkčnosti vzhledem ke stabilitě systému.



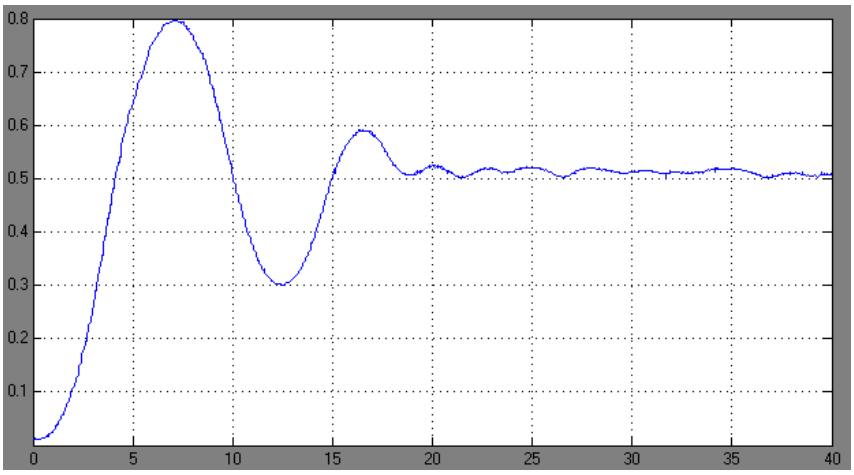
Obr. 6.2a hlavní schéma v Simulinku pro řízení reálného modelu



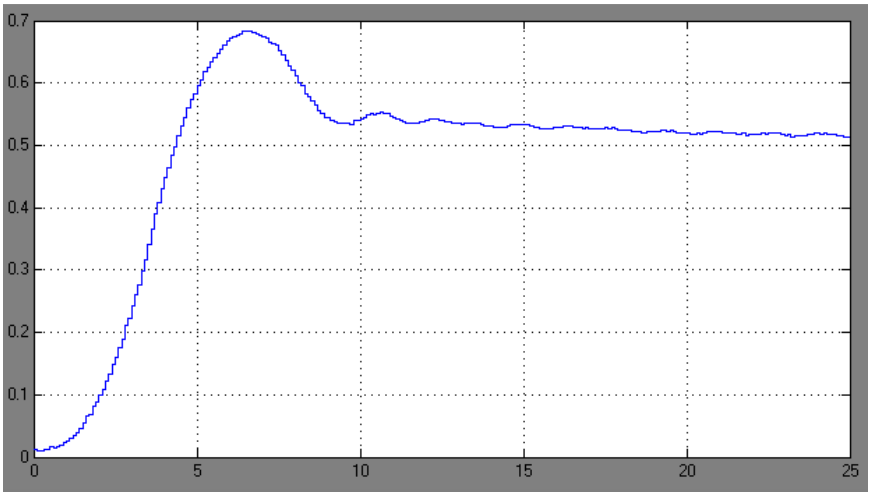
Obr. 6.2b Vnořené schéma funkčního bloku kulička na tyči

Vzhledem k níže provedeným testům a použité elektronice v modelu kulička na tyči, kde je servopohon řízen PWM signálem s frekvencí 50Hz, byla vhodná perioda pro řízení stanovena na právě periodu PWM signálu, čili 20ms. Dostatečná perioda řízení pak byla stanovena na 100ms.

Řízení modelu PSD regulátorem s parametrem řízení T=20ms:



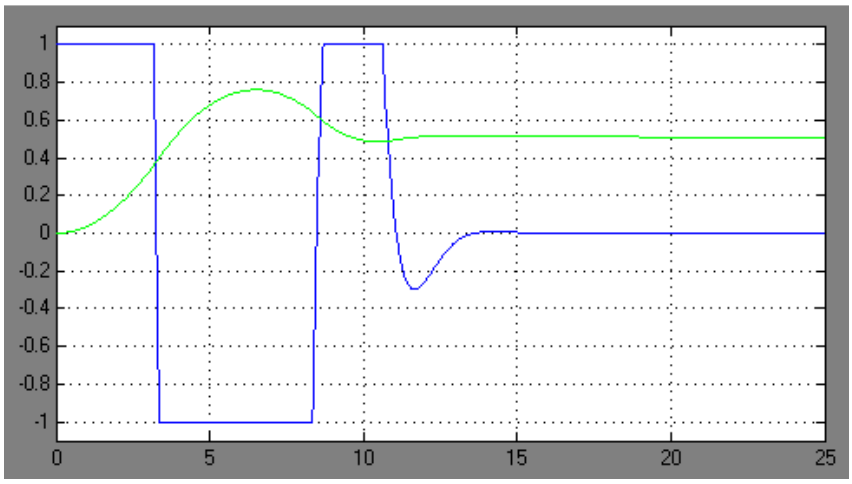
Obr. 6.2c Výstup reálné soustavy s PSD regulátorem s parametrem řízení T=20ms



Obr. 6.2d Výstup reálné soustavy s PSD regulátorem2 s parametrem řízení T=20ms

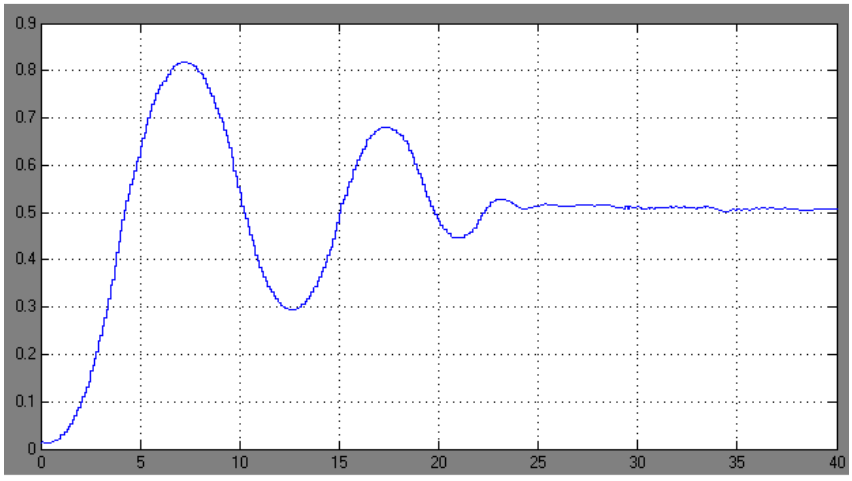
Na Obr. 6.2c můžeme sledovat značně odlišný výstup regulované soustavy oproti simulaci se shodným časem periody řízení. Musely se zde promítnout odlišnosti mezi přenosem simulované a reálné soustavy. Asi největší odlišnost reálného modelu od simulované soustavy je neschopnost modelu přesně sledovat akční veličinu. Servomotor, který ovládá ramenem naklonění tyče s kolejničkou, nedokáže měnit skokově žádanou výchylku, vzniká tak zpoždění

akční veličiny. Výše testovaný regulátor se saturací akční veličiny bohužel vede ke skokovým změnám na vstupu do soustavy, jak lze pozorovat na simulaci průběhu akční veličiny (Obr.6.2e) u spojitého PID regulátoru se saturací akční veličiny. Tento nešvar lze v tomto případě kompenzovat zvýšením derivační složky na dvojnásobek, jak je znázorněno na obr. 6.2d. Dojde tak sice k dřívějším změnám náklonu tyče a soustava již nekmitá, výsledná diference je, ale již tak vysoká, že šum vyskytující se ve snímané veličině napětí ze soustavy má za následek neustálý nemalý pohyb ramene z jedné strany na druhou. Zlepšení by nastalo filtrací snímané výstupní veličiny modelu.



*Obr. 6.2e Průběh u(t)*100 (modře) s omezením akční veličiny u PID regulátoru*

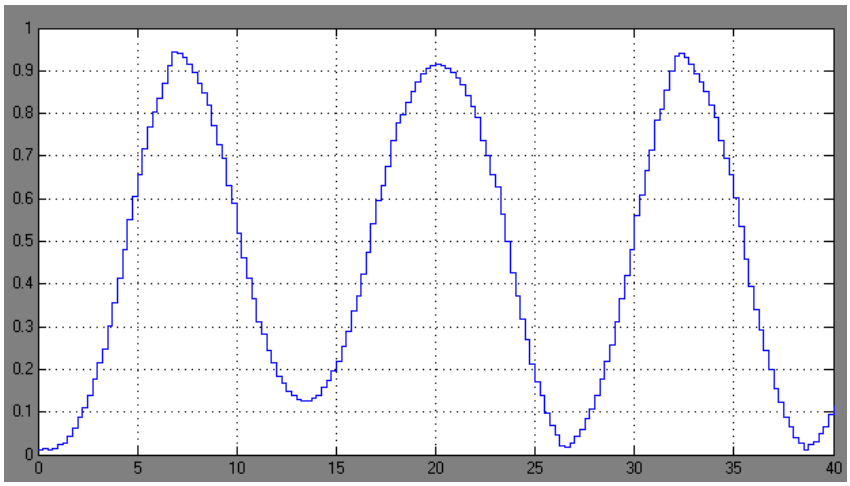
Řízení modelu PSD regulátorem s parametrem řízení T=100ms:



Obr. 6.2f Výstup reálné soustavy s PSD regulátorem s parametrem řízení T=100ms

Takový výsledek není sice vhodný pro řízení tohoto modelu, ale perioda 100ms dostačuje ke stabilitě systému.

Řízení modelu PSD regulátorem s parametrem řízení T=333ms:



Obr. 6.2g Výstup reálné soustavy s PSD regulátorem s parametrem řízení T=333ms

V tomto případě řízení s již tak velkou periodou řízení nemá smysl, protože výstupní veličina kmitá a soustava je tak nestabilní. Platí zde to samé jako v předcházející kapitole s T=100ms, avšak měření reakce regulované soustavy se zvýšenou derivační složkou na dvojnásobek neproběhla.

6.3 Testování RT vlastností vybraných platforem řízení

Pro odvozené parametry řízení T= 20-100ms, simulované a testované v předcházející kapitole, je vhodné se ujistit, jestli vybrané platformy pro řízení jsou schopny poskytnout kvality spojené se stabilitou řízené soustavy. Provede se tedy pár nativních testů ohledně latence a časové nestability (pojmy jsou vysvětleny v kapitole 4.1).

Protože je RT rozšíření Xenomai nainstalovaný jak na PC s DAQ, tak na Notebooku, který bude akční zásah počítat a zasílat zpět po LAN síti na PC, testy proběhnou na PC i na notebooku hned dvakrát a to pro zatížený a nezatížený systém. Zatížení systému bude provedeno kopírováním souboru.

Dobré RT vlastností systému na notebooku sice nejsou až tak zásadní, protože periodicky bude úloha spuštěna na PC. Výsledky budou přinejmenším zajímavé, protože Xenomai není pro

procesor na notebooku optimalizován, respektive nainstalovala se na něj verze zkompilovaná pro procesor typu *Intel Pentium 4* místo *Core 2 Duo.* I přes to musí test latence vykazovat jistých akceptovatelných časů, pokud chceme použít API pro RT rozšíření Xenomai a pouštět na notebooku RT tasky.

6.3.1 Testování latence

Tento test ukáže, za jakou dobu dokáže OS přistoupit k vykonání obsluhy události od jejího vyvolání v uživatelském prostoru (User Space). Test se spouští z konzole a po vykonání, v našem případě, deseti tisíců obsluh událostí v periodě volání 100µs se vypíše jeden řádek s výsledky měření v mikrosekundách:

- lat min - minimální čas latence
- lat avg - průměrný čas latence
- lat max - maximální čas latence
- overrun - počet případů kdy byla latence vyšší než vykonávaná perioda volání
- lat best - nejnižší latence v celém testu
- lat worst - nejvyšší latence v celém testu

Výsledky tesu se také může využít pro korekci průměrné latence v plánovači RT úloh.
[25]

Výstup z konzole nezatíženého Xenomai na notebooku:

```
root@loid:/usr/xenomai/share/xenomai/testsuite/latency# ./run
*
*
* Type ^C to stop this application.
*
*
== Sampling period: 100 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:02  (periodic user-mode task, 100 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      0.135|      0.838|     20.459|       0|     0|      0.135|     20.459
RTD|      0.263|      0.862|     20.722|       0|     0|      0.135|     20.722
RTD|      0.360|      0.857|     20.790|       0|     0|      0.135|     20.790
RTD|      0.137|      0.847|     20.651|       0|     0|      0.135|     20.790
RTD|      0.224|      0.826|     20.444|       0|     0|      0.135|     20.790
RTD|      0.263|      0.843|     20.700|       0|     0|      0.135|     20.790
RTD|      0.374|      0.854|     20.692|       0|     0|      0.135|     20.790
RTD|      0.229|      0.847|     20.691|       0|     0|      0.135|     20.790
RTD|      0.367|      0.850|     20.723|       0|     0|      0.135|     20.790
```

Výstup z konzole zatíženého Xenomai na notebooku:

```
root@loid:/usr/xenomai/share/xenomai/testsuite/latency# ./run
*
*
* Type ^C to stop this application.
*
*
== Sampling period: 100 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01  (periodic user-mode task, 100 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      0.390|      0.676|      2.111|       0|     0|      0.390|      2.111
RTD|      0.405|      0.647|      1.996|       0|     0|      0.390|      2.111
RTD|      0.313|      0.671|     15.535|       0|     0|      0.313|     15.535
RTD|      0.155|      0.591|      1.963|       0|     0|      0.155|     15.535
RTD|      0.334|      0.678|      2.172|       0|     0|      0.155|     15.535
RTD|      0.348|      0.679|     15.498|       0|     0|      0.155|     15.535
RTD|      0.129|      0.639|      2.668|       0|     0|      0.129|     15.535
RTD|      0.387|      0.646|      2.260|       0|     0|      0.129|     15.535
RTD|      0.359|      0.604|      2.268|       0|     0|      0.129|     15.535
```

U zatíženého systému lze pozorovat větší nestabilita nejvyšší latence, a to i když jsou průměrné latence nižší než u testu s nezatíženým systémem. Xenomai se tak pravděpodobně snaží o autokorekci v plánovači RT úloh.

Výstup z konzole nezatíženého Xenomai na PC (v prostoru jádra):

```
root@pce228a:/usr/xenomai/bin# ./latency -t 1
== Sampling period: 100 us
== Test mode: in-kernel periodic task
== All results in microseconds
warming up...
RTT|  00:00:01  (in-kernel periodic task, 100 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      1.358|      1.539|      2.422|       0|     0|      1.358|      2.422
RTD|      1.391|      1.650|      3.257|       0|     0|      1.358|      3.257
RTD|      1.283|      1.668|      4.193|       0|     0|      1.283|      4.193
RTD|      1.354|      1.616|      3.473|       0|     0|      1.283|      4.193
RTD|      1.343|      1.641|      2.780|       0|     0|      1.283|      4.193
RTD|      1.444|      1.700|      6.117|       0|     0|      1.283|      6.117
RTD|      1.344|      1.669|      2.533|       0|     0|      1.283|      6.117
RTD|      1.366|      1.704|      3.356|       0|     0|      1.283|      6.117
RTD|      1.355|      1.484|      2.812|       0|     0|      1.283|      6.117
RTD|      1.423|      1.702|      3.290|       0|     0|      1.283|      6.117
```

Výstup z konzole zatíženého Xenomai na PC:

```
root@pce228a:/usr/xenomai/bin# ./latency
== Sampling period: 100 us
== Test mode: periodic user-mode task
== All results in microseconds
warming up...
RTT|  00:00:01  (periodic user-mode task, 100 us period, priority 99)
RTH|----lat min|----lat avg|----lat max|-overrun|---msw|---lat best|--lat worst
RTD|      3.086|      3.319|      9.736|       0|     0|      3.086|      9.736
RTD|      2.445|      3.459|      9.158|       0|     0|      2.445|      9.736
RTD|      2.426|      3.500|      9.295|       0|     0|      2.426|      9.736
RTD|      2.949|      3.325|      9.235|       0|     0|      2.426|      9.736
RTD|      3.004|      3.310|      9.565|       0|     0|      2.426|      9.736
RTD|      2.990|      3.467|      9.406|       0|     0|      2.426|      9.736
RTD|      3.011|      3.480|      9.426|       0|     0|      2.426|      9.736
```

Z důvodu zjistit kde až jsou schopnosti Xenomai vytvářet nejrychlejší obsluhu události (nejnižší latence) na testovaném hardware, byl proveden test při nezatíženém systému v prostoru jádra. Xenomai je zřejmě pro použitý procesor dobře optimalizován, protože při zatíženém systému stoupla průměrná latence o necelé 3µs, u nejhorší naměřené pak stoupla přibližně o 4µs.

Obecně u zatížených systémů šlo v testu pozorovat zpoždění výpisu na obrazovku, grafické prostředí operačního systému se dostalo prioritně do pozadí, testovací *task* dostával s prioritou 99 více procesorového času.

Takové testy můžeme prohlásit za více než dostatečné pro využití OS Linux s Xenomai k regulaci zvoleného modelu i na základě např. přepínání vláken bez časového hundleru, protože průměrná zpoždění jsou řádově 10000 krát nižší než např. perioda řízení T=20ms, která byla označena v minulých kapitolách za vhodnou pro daný model.

6.3.2 Test časové nestability (jitter):

Tento test je pro opodstatnění vhodnosti testovaného OS s RT vlastnostmi pro využití k řízení modelu kuličky na tyči důležitější než test latence. Důvod je ten, že se v programu nebude nacházet časté spouštění RT úloh, ale zcela jistě poběží jedna s vykonávaným časovým přerušením (od Time Hundleru) v cyklu odvozeném od periody řízení soustavy.

Zjištění odchylky od chtěné periody cyklu (časové chvění) nám pak prozradí, jestli je takový systém vhodný pro řízení a regulaci kuličky na tyči, kde je dostačující perioda pro udržení stability systému vzorkování pod 100ms, vhodná je pak kolem 20ms. (vycházíme z kapitoly 6.2)

Výstup z konzole nezatíženého Xenomai na notebooku:

```
root@loid:/home/loid/trivial-periodic# ./trivial-periodic
Perioda = 10 ms

cas od minuleho cyklu: 10.014558ms, chyba: 14.558micsec
cas od minuleho cyklu: 10.000518ms, chyba: 0.518micsec
cas od minuleho cyklu: 9.987429ms, chyba: 12.571micsec
cas od minuleho cyklu: 10.012352ms, chyba: 12.352micsec
cas od minuleho cyklu: 9.999537ms, chyba: 0.463micsec
cas od minuleho cyklu: 9.987192ms, chyba: 12.808micsec
cas od minuleho cyklu: 10.014378ms, chyba: 14.378micsec
cas od minuleho cyklu: 9.998849ms, chyba: 1.151micsec
cas od minuleho cyklu: 9.986557ms, chyba: 13.443micsec
cas od minuleho cyklu: 10.013603ms, chyba: 13.603micsec
```

Výstup z konzole zatíženého Xenomai na notebooku:

```
root@loid:/home/loid/trivial-periodic# ./trivial-periodic
Perioda = 750 ms

cas od minuleho cyklu: 750.006662ms, chyba: 6.662micsec
cas od minuleho cyklu: 750.015111ms, chyba: 15.111micsec
cas od minuleho cyklu: 749.984885ms, chyba: 15.115micsec
cas od minuleho cyklu: 750.000553ms, chyba: 0.553micsec
cas od minuleho cyklu: 749.999805ms, chyba: 0.195micsec
cas od minuleho cyklu: 749.999692ms, chyba: 0.308micsec
cas od minuleho cyklu: 750.014660ms, chyba: 14.660micsec
cas od minuleho cyklu: 749.986396ms, chyba: 13.604micsec
cas od minuleho cyklu: 749.999812ms, chyba: 0.188micsec
cas od minuleho cyklu: 750.000339ms, chyba: 0.339micsec
cas od minuleho cyklu: 749.999045ms, chyba: 0.955micsec
cas od minuleho cyklu: 750.014623ms, chyba: 14.623micsec
cas od minuleho cyklu: 749.985810ms, chyba: 14.190micsec
cas od minuleho cyklu: 750.013507ms, chyba: 13.507micsec
cas od minuleho cyklu: 749.986309ms, chyba: 13.691micsec
```

Testy časové nestability na notebooku nedopadly sice tak dobře jako na PC, pro který je Xenomai optimalizovaný, ale vzhledem k nejmenší plánované periodě řízení 20ms jsou rozdíly periody řádově v tisícinách.

Výstup z konzole nezatíženého Xenomai na PC:

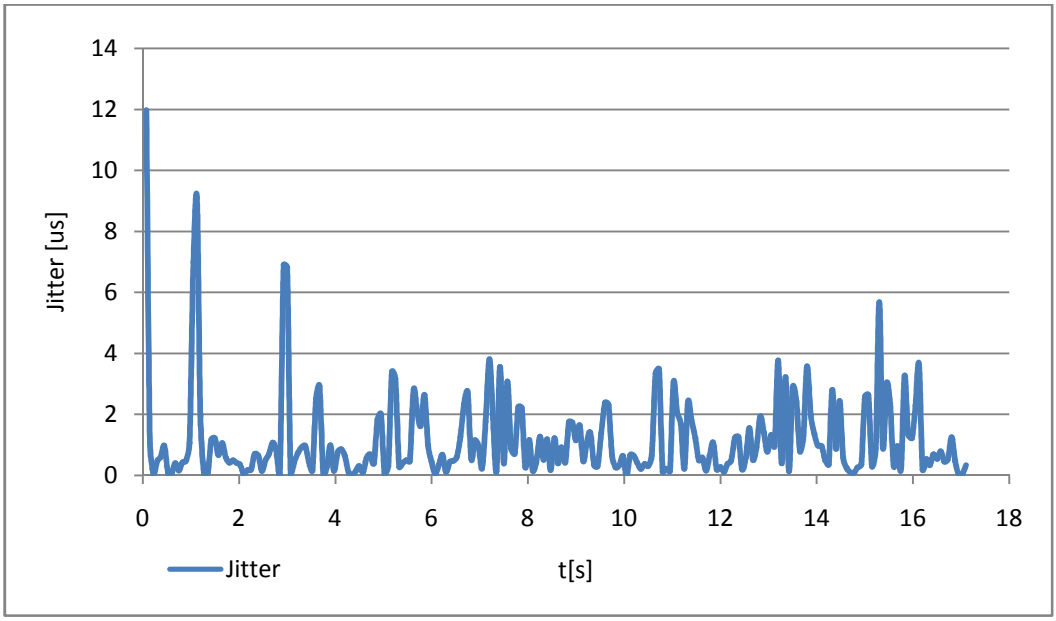
```
root@pce228a:/home/loid/DP/xehu#./trivial-periodic
Perioda = 75 ms

Perioda: 75.011972ms, chyba: 11.972micsec
Perioda: 74.998959ms, chyba: 1.041micsec
Perioda: 74.999942ms, chyba: 0.058micsec
Perioda: 75.000474ms, chyba: 0.474micsec
Perioda: 75.000609ms, chyba: 0.609micsec
Perioda: 74.999027ms, chyba: 0.973micsec
Perioda: 74.999920ms, chyba: 0.080micsec
```

Výstup z konzole zatíženého Xenomai na PC

```
root@pce228a:/home/loid/DP/xehu#./trivial-periodic
Perioda = 75 ms

perioda 10.001163ms, chyba je 1.163micsec
perioda 9.990690ms, chyba je 9.310micsec
perioda 10.007435ms, chyba je 7.435micsec
perioda 9.993056ms, chyba je 6.944micsec
perioda 9.999407ms, chyba je 0.593micsec
perioda 10.007237ms, chyba je 7.237micsec
perioda 9.992868ms, chyba je 7.132micsec
perioda 10.000836ms, chyba je 0.836micsec
perioda 10.005996ms, chyba je 5.996micsec
perioda 9.993487ms, chyba je 6.513micsec
perioda 9.999475ms, chyba je 0.525micsec
perioda 10.008365ms, chyba je 8.365micsec
```

Testy časové nestability na PC dosáhly přibližně dvakrát menších hodnot než na notebooku, zde jde pozorovat optimalizaci Xenomai pro používaný hardware na PC. Vzhledem k nejmenší plánované periodě řízení 20ms jsou rozdíly periody řádově v tisících při zatíženém systému.

Test jittru v čase v grafu 6.3.2 neproběhl na zatíženém systému kopírováním, ale během komunikace s klientem. Výsledky jsou stejně jako u notebooku pro danou úlohu regulace akceptovatelné i pří komunikaci přes protokol UDP/IP.



Graf 6.3.2 Průběh časové nestability na PC při komunikaci s klientem na notebooku

6.3.3 Testování vlastností RTnet

Testy proběhly pouze na lokální adrese samotného počítače (localhost), protože se nepovedlo rozběhnout komunikaci mezi dvěma účastníky sítě. Ukázalo se, že nastavování pomocí konfiguračních souborů RTnet nejsou vůbec triviální, načež pak nahrání vhodných modulů a instalace RT ovladačů pro sítové karty nemohlo být dokončeno z důvodu nedostatečného hardwarového vybavení na notebooku. Problémem byla pro RTnet nepodporovaná síťová karta 82567LM GIGABIT NET.

Po dlouhém tápání jak správně kompilovat zdrojový kód se podařilo zkompilovat a nastavit konfigurační soubory tak, aby komunikace probíhala v rámci zařízení RTLoop.

Výstup z konzole notebooku, Test zpoždění Pink (LocalHost):

```
root@loid:/usr/local/rtnet/sbin# ./rtnet start
ioctl: No such device
ioctl: No such device
ioctl: No such device
ioctl: No such device
ioctl (add): No such device
ioctl (add): No such device
ioctl (add): No such device
vnic0: ERROR while getting interface flags: Takove zarizeni neexistuje
SIOCSIFADDR: Takove zarizeni neexistuje
vnic0: ERROR while getting interface flags: Takove zarizeni neexistuje
Waiting for all slaves...ioctl: No such device
ioctl: No such device
```

Nastavení konfiguračního souboru, při spuštění sikriptu zjisti absenci některých zařízení

```
root@loid:/usr/local/rtnet/sbin# ./rtifconfig
rtlo      Medium: Local Loopback
          IP address: 127.0.0.1
          UP LOOPBACK RUNNING  MTU: 1500
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

Výpis zařízeni pro RT komunikaci, pouze LocalHost.

```
root@loid:/usr/local/rtnet/sbin# ls
rtcfg  rtifconfig  rtiwconfig  rtnet  rtping  rtroute  tdmacfg
root@loid:/usr/local/rtnet/sbin# ./rtping -erg
Usage:
       rtping [-c count] [-i interval] [-s packetsize] <addr>
root@loid:/usr/local/rtnet/sbin# ./rtping -c 5 -i 10 127.0.0.1
Real-time PING 127.0.0.1 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 time=4.3 us
64 bytes from 127.0.0.1: icmp_seq=2 time=1.6 us
64 bytes from 127.0.0.1: icmp_seq=3 time=1.7 us
64 bytes from 127.0.0.1: icmp_seq=4 time=3.1 us
64 bytes from 127.0.0.1: icmp_seq=5 time=1.5 us
```

```
--- 127.0.0.1 rtping statistics ---
5 packets transmitted, 5 received, 0% packet loss
   worst case rtt = 4.3 us
```

test RT komunikace na lokální smyčce Notebooku

Výstup z konzole notebooku, test odezvy Sender-responder (LocalHost):

Console1:

```
root@loid:/usr/src/rtnet-0.9.12/examples/xenomai/posix# ./rtt-responder -l
127.0.0.1
```

spuštění responderu, ten čeká na sender.

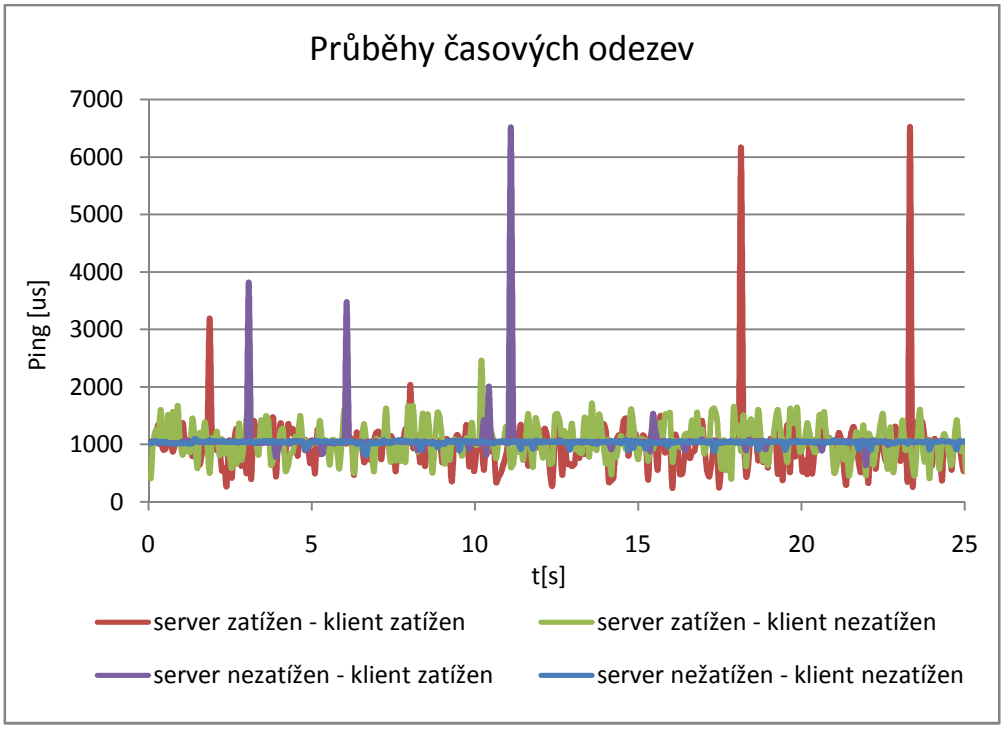
Concole2:

```
root@loid:/usr/src/rtnet-0.9.12/examples/xenomai/posix# ./rtt-sender -c
10000 -l 127.0.0.1
destination ip address: 127.0.0.1 = 0100007f
local ip address: 127.0.0.1 = 0100007f
cycle: 10000 us
WARNING: ioctl(RTNET_RTIOC_EXTPOOL): Invalid argument
127.0.0.1        19.416 us, min=   17.041 us, max=   40.717 us, count=239
^Cshutting down
terminating transmitter thread
terminating receiver thread
```

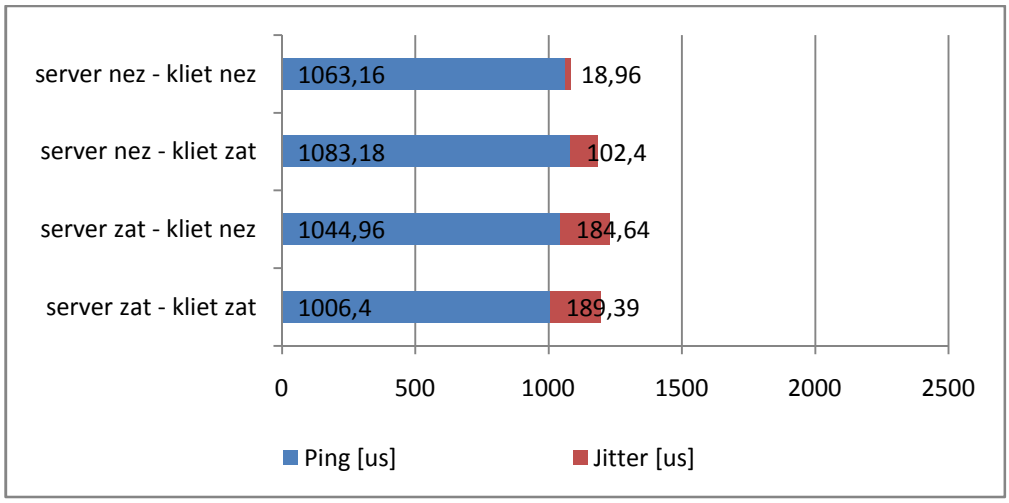
spuštění senderu, odesláno a přijato zpět 239x s průměrnou odezvou 19.4us.

6.3.4 Testy UDP/IP komunikace

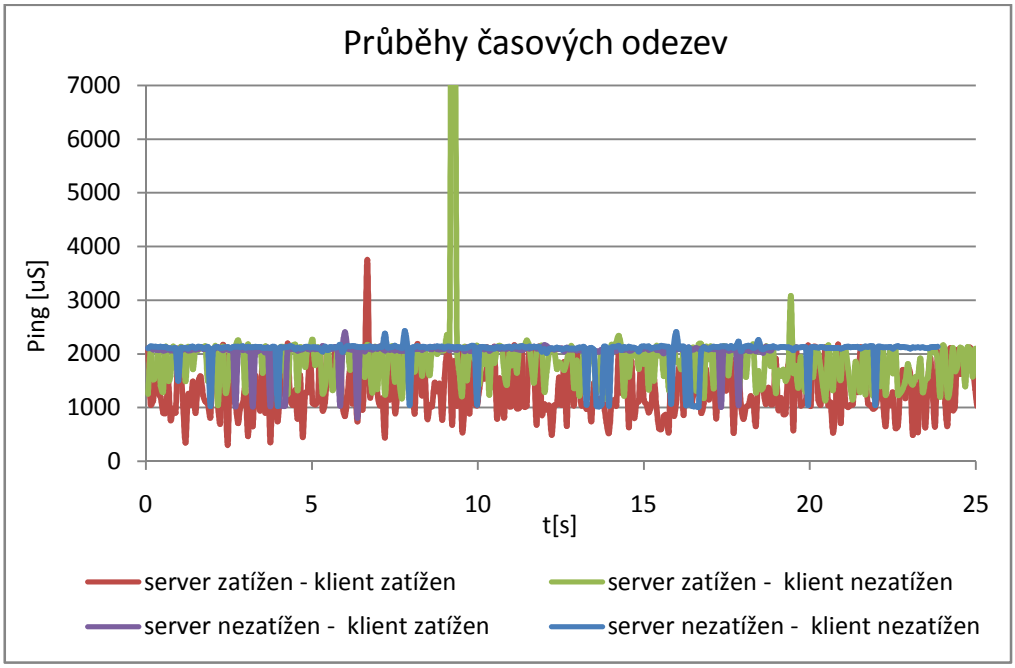
Výhledy na komunikaci s použití protokolu z řad RTnet nevypadaly slibně, proto proběhly testy komunikace přes protokol UDP/IP. Zatížení sítě bylo provedeno stahováním dat z internetu rychlostí 3.5MB/s. Samotné měření proběhlo ve vytvořených aplikacích server, klient.



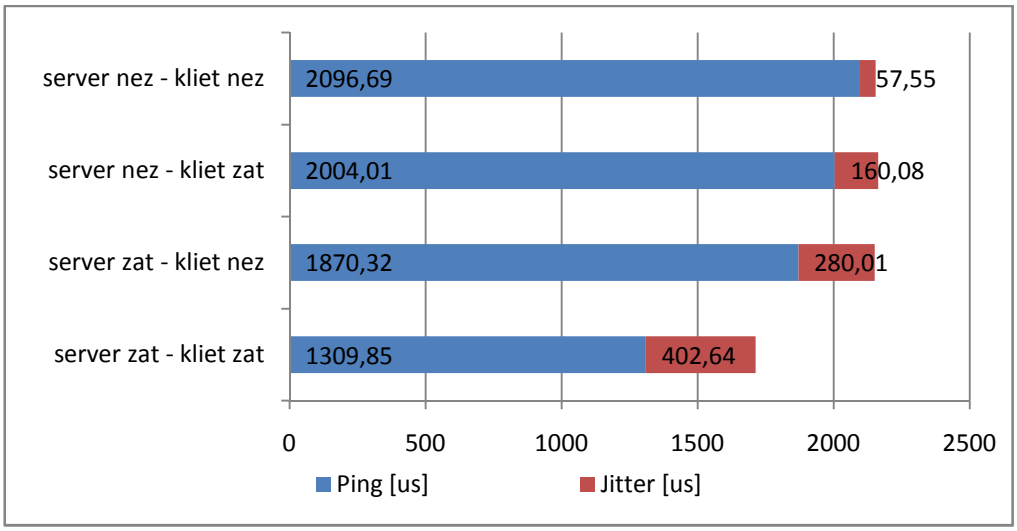
Graf 6.2.3a Průběhy komunikací mezi serverem PC-Xenomai a klientem Notebook-Xenomai



Graf 6.2.3b Průměrné časy pingu a jitteru mezi serverem PC-Xenomai a klientem Notebook-Xenomai

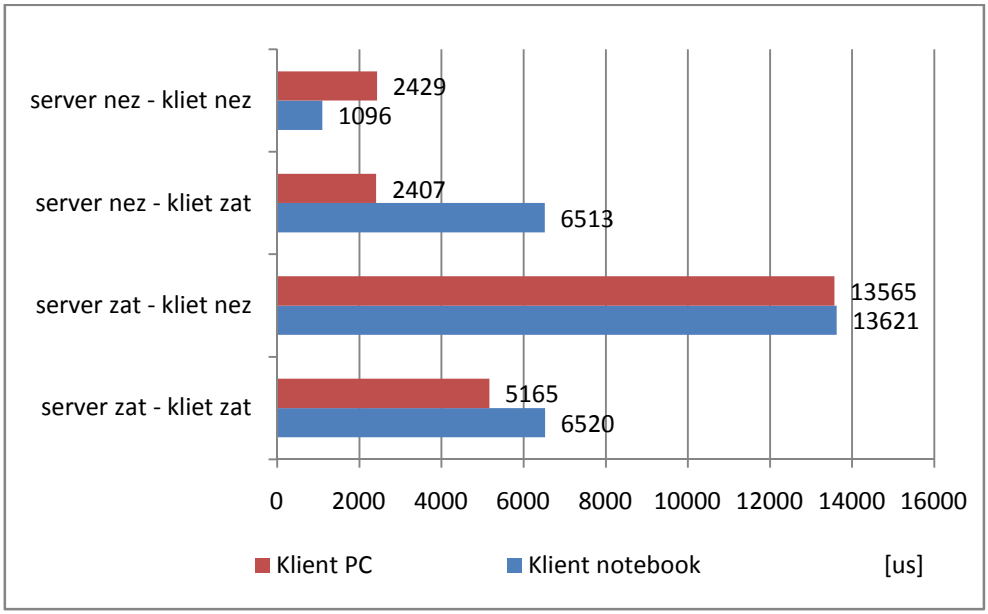


Graf 6.2.3c Průběhy komunikací mezi serverem PC-Xenomai a klientem PC-Linux 2.6



Graf 6.2.3d Průměrné časy pingu a jitteru mezi serverem PC-Xenomai a klientem PC-Linux 2.6

Z grafů je zřejmé, že se časový nedeterminismus (jitter) daný kolísáním od průměrné hodnoty odezvy (pingu) zvedá se zatížením sítě. Způsob jakým se zatěžovala síť (stahování dat rychlostí 3,5Mb/s) však zatěžovala i operační systém. Tento jev lze pozorovat na téměř dvounásobné odezvě u PC jako klienta oproti notebooku. Důvody nižší odezvy bez zatížení jsou jednoznačně dány jednak vyšším výkonem hardware a výhodnějšímu plánování procesů v Xenomai, který má obecně vyšší prioritu než běžné procesy v Linuxu.



Graf 6.2.3e Nejvyšší změřený ping ve všech variantách jednotlivých účastníku UDP/IP komunikace

Nejvyšší naměřená odezva za dobu komunikace 20-90s byla 13.62ms. Byly zaznamenány prodlevy, které činily až desetinásobek průměru měřených odezev. Pro Hard Real-time aplikace je tedy jednoduchá UDP/IP komunikace nespolehlivá i při použití jednoduchého konceptu producent-konzument a RTOS, pokud požadujeme deterministické chování při zatížené síti.

Jestli bychom chtěli provozovat regulaci vybraného modelu kuličky na tyči pomoci vzdáleného řízení přes UDP/IP protokol na výše testovaném hardware, musila by být stanovena perioda řízení na základě dlouhodobého testování kvality komunikace také vzhledem k chybám. Pokud by při testování nedošlo k většímu pingu, jako bylo doposud naměřeno nebo chybám, dala by se perioda řízení stanovit na 20ms. Tak by bylo dosaženo vždy stejného chování modelu při jeho řízení. Pro ještě menší periodu řízení pod 13ms by se dal do jisté míry systém označit jako Soft Real-time a v případě nedodržení deadline by řízení probíhalo jinak popřípadě s větším časovým úsekem ustálení než u silně deterministického řízení a podmínky, které by pak vedly k udržení stability systému by se daly převést na Soft Real-time požadavky.

7 Implementace části měření a akčního zásahu pomocí vhodné karty DAQ

7.1 Návrh algoritmu pro vykonávání periodické RT úlohy

Způsob periodického volání kódu, jako bylo použito v testu časové stability (5.3.2) pomocí upraveného programu trivial-periodic je v API Xenomai prováděno pomocí inicializace *RT_task*, jeho vytvořením a následným zavoláním s vykonáním příslušné funkce. V těle provedené funkce se nachází inicializace *wait* funkce, která je umístěná v smyčce *while*. Kód je naznačen v příkladu a perioda použita ve funkci *Wait* je nastavena na 10ms.

```
RT_TASK demo_task;
void demo(void *arg)
{
       /* Arguments: &task (NULL=self),
        *            start time,
        *            period (here: 1 s)
        */
       rt_task_set_periodic(NULL, TM_NOW, period);

       while (1) {
              rt_task_wait_period(NULL);
              */ Tělo periodicky vykonávaného kódu. */
              }
}

int main(int argc, char* argv[])
{
       signal(SIGTERM, catch_signal);
       signal(SIGINT, catch_signal);

       /* volání virtuálního prostoru v RAM pro swapping */
       mlockall(MCL_CURRENT|MCL_FUTURE);

       /* Argumenty: &task,
        *            name,
        *            stack size (0=default),
        *            priority,
        */            mode (FPU, start suspended, ...)

       rt_task_create(&demo_task, "trivial", 0, 99, 0);

       /* Arguments: &task,
        *            task function,
        *            function argument
        */

       rt_task_start(&demo_task, &demo, NULL);
       pause();
       rt_task_delete(&demo_task);
       return(0);
}
```

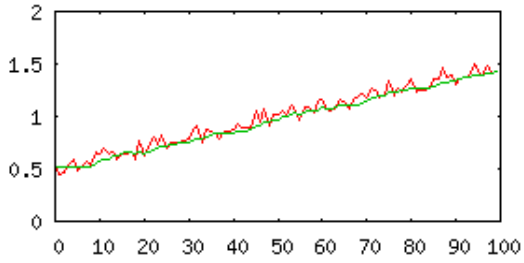
7.2 Implementace filtru měřené veličiny

Z důvodu velkého šumu snímané výstupní veličiny (napětí) modelu soustavy, byla navrhnuta softwarová filtrace. Zdroj šumu nebyl zjištěn, buď jim byly operační zesilovače v modelu, nebo měřící karta v PC.

Plovoucí průměr snímané veličiny je jednoduchý způsob jak z průměru několika posledních hodnot snímané veličiny vylepšit vlastnosti regulace.

Příklad:

```
sum = 0.0;
  for (i = n-k; i < n; i++)
     sum += y[i];
  avg = sum/k;
```



Obr. 7.2 Plovoucí průměr pro k=8 [23]

I plovoucí průměr se může počítat rekurzivně a složitost výpočtu tedy nemusí záviset na délce průměrovacího okna *k*:

```
avg = (avg*k - y[n-k] + y[n]) / k;
```

V tomto vztahu reprezentuje člen *avg*k* součet posledních *k* měření. V dalším kroku od tohoto součtu tedy odečteme nejstarší měření *x[n-k]* a naopak přičteme nové měření *x[n]*, čímž získáme součet posledních *k* měření. Vydělením hodnotou *k* tedy získáme průměr z posledních *k* měření. Rekurzivní výpočet plovoucího průměru se může přepsat jako součet posledního průměru a určité korekce:

```
avg = avg + 1/k * (y[n] - y[n-k]);
```

Váha určující vliv korekční složky je nyní konstantní a odpovídá převrácené hodnotě délky průměrujícího okna. Velikost korekce je dána rozdílem nejnovějšího a nejstaršího měření. Plovoucí průměr tedy prokládá vodorovnou přímku posledními *k* měřeními (viz obrázek). Rekurzivní definicí výpočtu jsme opět získali konstantní složitost (tj. složitost nezávislou na délce průměrovacího okna). Stále si ale musíme pamatovat všech posledních *k* měření, která jsou zahrnuta do plovoucího průměru.
[23]

Délka průměrovacího okna *k* byla stanovena v závislosti na velikosti derivační konstanty, šumu a rychlosti snímání (T=10ms) na *k*=50. Velikost průměrovacího okna má za následek posunutí filtrovaného signálu v čase, proto bylo nutností zvolit tuto konstantu vhodně tak, aby se tendence kmitat ramenem modelu z důvodu vysoké reakce na derivaci šumu potlačila, ale zároveň nemůže být příliš vysoká z důvodu následně posunuté akční veličiny v požadovaném čase. Algoritmus byl vytvořen tak aby nebyla nutnost si pamatovat všechny naměřené hodnoty, protože v opačném případě by došlo k přetečení datového typu.

7.3 Implementace PSD algoritmu

Způsobů jak vytvořit PID, respektive PSD algoritmus v jazyce C je několik, protože v simulaci bylo dosaženo dobrých výsledků s doplněním o Anti-Wind-Up efekt, zvolený algoritmus je spočívá v součtu jednotlivých P-proporcionální, D-diferenční a S-sumační složek. Omezení jedné složky, v tomto případě S-sumační složky na hodnotu saturace ramene dělenou šesti, pak v jazyce C není složité a vypadá následovně:

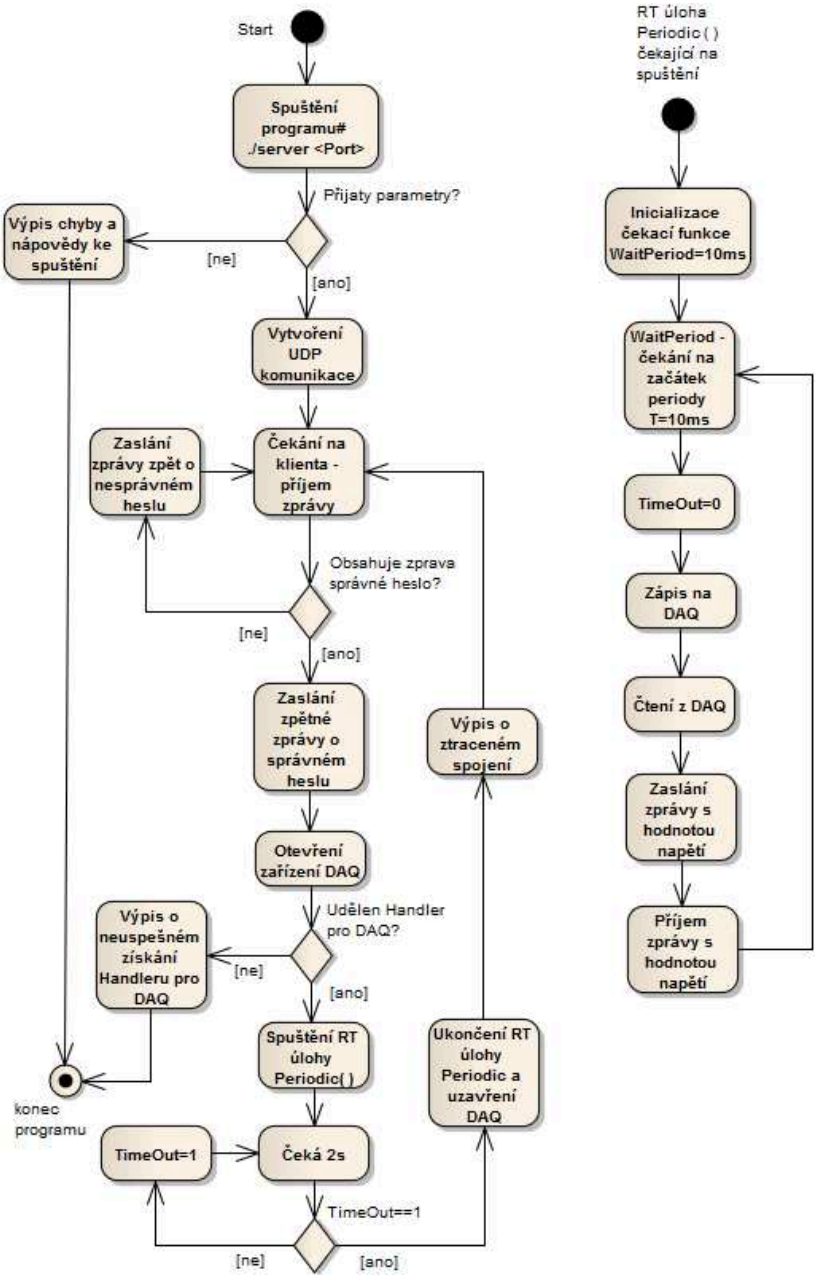
```
//vypocet regulacni odchylky
e0=sp-pvFil;

//---vypocteni akcniho zasahu---//
P=e0*Kp;
S=e1*T*Ki;
D=(e0*(Kd/T))-(e1*(Kd/T));
suma=suma+S;

//omezeni windup efektu
if(suma>0.01/6) suma=0.01/6;
if(suma<(-0.01/6)) suma=(-0.01/6);

//socet slozek PSD a upraveni na snesitelnou mez pro model
u=(P+suma+D)/5;

//saturace ramene
if(u>0.01) u=0.01;
if(u<(-0.01)) u=(-0.01);


//prepis odchylek do prislusnych kroku
e2=e1;
e1=e0;
```

Protože servomechanismus nemůže rychle reagovat na skokové změny akční veličiny, jak je naznačeno v kapitole 6.2 o ověřování periody řízení, je celková hodnota této akční veličiny snížená na jedno pětinu. Výsledné průběhy akční veličiny nebudou již mít tak skokový průběh a servomotor bude moci sledovat svou výchylkou požadovanou hodnotu.

8 Implementace regulátoru v rámci NCS sítě a vizualizace úlohy v PC.

Vytvořit deterministickou komunikaci, která by stála na protokolu RTnet by sice bylo možné, ale protože výše zmíněné testy RT komunikací proběhly pouze v rámci Local Hostu a ne mezi fyzicky vzdálenými účastníky, byla pro splnění konceptu NCS vytvořena náhrada komunikace protokolem UDP/IP.

8.1 Návrh serveru



Obr. 8.1 Aktivity diagram chování serveru

Na Obr. 8.1 je naznačeno chování serveru pomocí UML diagramu typu Ativity. Server má ve vzdáleném řízení roli udržovatele přesné periody řízení, ve které provádí příslušná měření a akční zásahy zápisem na DAQ kartu. PC na němž server běží, je tedy fyzicky u řízeného modelu a komunikuje pomocí zpráv se vzdáleným klientem, který je v roli regulátoru. Použitý protokol UDP/IP a komunikace na něm byla testována v kapitole 6.3.4.

Co se týče komunikace, server obsahuje nezbytné části pro navázání spojení pomocí hesla a jeho znovu navázání při ztrátě spojení. Server tak může pracovat ve dvou různých stavech, stavu čekání na klienta a stavu periodického zápisu a čtení z DAQ.

Server používá navrhnutá řešení v kapitole 7 a je doplněn o komunikaci a náležitosti s tím spojené, které jsou dále popsané.

Program server obsahuje dvě vlákna:

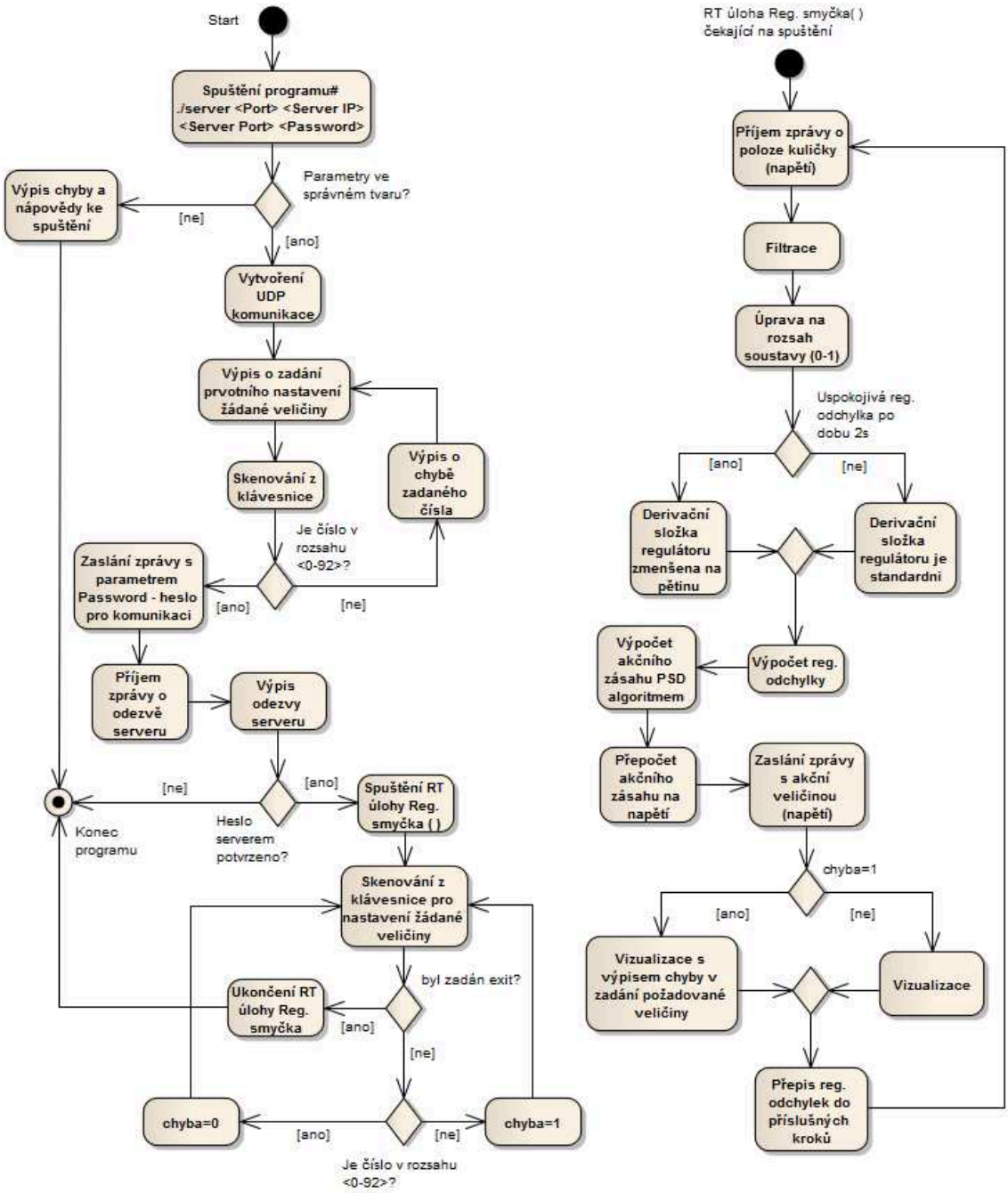
Hlavní:

- Startem programu se začne vykonávat funkce Main (), které má vstupní parametr Lokální port. Klient, který se bude připojovat, musí znát IP adresu a Port, aby se mohla navázat komunikace.
- Inicializuje UDP komunikaci
- Ověřuje správnost hesla pro přechod do stavu řízení
- Spouští RT vlákno s funkcí Periodic
- Kontroluje ztrátu spojení pomocí pozorovatele, který v tomto případě ukončí RT tast a přepne server do stavu čekání na klienta

RT task Periodic:

- Vlákno je typu RT s prioritou 95
- Prvotně je toto RT vlákno spuštěno z hlavního vlákna
- Vykonává periodicky s T=10ms funkcí zápisu a čtení na DAQ kartu.
- Posílá klientovi zprávy o poloze kuličky (napětí)
- Přijímá od klienta zprávy o akčním zásahu (napětí)
- Dokáže si měřit čas mezi odesláním a příjmem packetů (ping) a čas periody. Ty pak vypisuje do konzole.

8.2 Návrh klienta



Obr. 8.2 Aktivity diagram chování klienta

Na Obr. 8.2 je naznačeno chování klienta pomocí UML diagramu typu ativity. Použitý protokol UDP a komunikace na něm byla testována v kapitole 6.3.4. Co se týče komunikace, klient jako první pošle zprávu s heslem o přístup k řízení modelu kulička na tyči serveru a pokud bude heslo správné, začne komunikace mezi nimi periodická komunikace. Klient zde hraje roli vzdáleného regulátoru s algoritmem PSD, a tedy přijímá informace o stavu soustavy a následně vypočítává akční zásah a zasílá ho zpět serveru.

Klient používá navrhnutá řešení v kapitole 7 a je doplněn o vizualizaci, komunikaci a náležitosti s tím spojené, které jsou dále popsané.

Program klient obsahuje dvě vlákna:

Hlavní:

- Spuštěním programu se začne vykonávat funkce Main, která má vstupní parametry: lokální port, IP adresu, server port, heslo pro řízení.
- Inicializuje UDP komunikaci.
- Obsahuje periodické skenování z klávesnice pro nastavení žádané hodnoty kuličky.
- Spouští RT vlákno s funkcí PSD smyčky.

RT task PSD smyčka:

- Vlákno je typu RT s prioritou 95.
- Prvotně je toto RT vlákno spuštěno z hlavního vlákna
- Přijímá zprávy od serveru o poloze kuličky.
- Zasílá zprávy serveru o akčním zásahu, vypočítaném pomocí PSD algoritmu.
- Filtruje na základě plovoucího průměru posledních padesát hodnot výstupní veličinu soustavy (polohu kuličky).
- Vykonává periodicky funkcí výpočtu akčního zásahu.
- Dokáže si na základě vhodných podmínek změnit konstanty regulátoru pro funkci jemnější regulace.

Klient také obsahuje jednoduchou vizualizaci v konzoli příkazového řádku. Ta znázorňuje pohyb kuličky pomocí horizontálního posunu znaku ‘o’ na jednom řádku, příslušní místo žádané hodnoty je vyznačeno znakem ‘X’. Mimo jiné se v terminálu zobrazují číselné hodnoty o stavu soustavy, které jsou: žádaná hodnota, aktuální (filtrovaná) hodnota polohy kuličky, akční veličina v hodnotě radiánů vynásobená stem a regulační odchylka.

Příklad vizualizace:

```
odezva serveru: Connect

zadana hodnota: 25, aktualni hodnota: 44.243773, akcni velicina: -1.000000
reg.odchylka: -19.243773

_____________________________________o_____________________________________
                     X

Zadej SP <0-92>:
```

8.3 Kompilace zdrojových kódu

Správné zkompilování zdrojových kódu v prostředí Open Source GNU/Linux sebou nese úpravy skriptů, zvaných makefile, a konfiguračních souborů, které jsou základním obsahem Patchů Xenomai a RTnet. Skripty, které obsahují cesty k potřebným souborům, jako jsou knihovny, headery a konfigurační soubory k správnému zkompilování kompilátorem, většinou GCC, jsou napsány pro konkrétní zdrojové kódy. Po doplnění zdrojového kódu o nové funkce již nemohou bez úprav skriptu byt zdrojové kódy správně zkompilovány. Mnoho skriptu není funkčních ani bez úprav zdrojového kódu a je třeba je doplnit o cesty k určitým knihovnám nebo změnit nějaké části skriptu. Další variantou je samotný příkaz GCC, který každý makefile obsahuje.

V příloze jsou obsazeny společně se zdrojovými kódy i změněné skripty a konfigurační soubory, které jsou nutné k jejich zkompilování do spustitelné podoby. Změny, které se musely provést pro jednotlivé zdrojové soubory, byly:

Server.c a klient.c:

- Křížový link (odkaz) na knihovnu pro DAQ karty lhudaqlib do systémové složky knihoven.
- Úprava souboru Makefile pro trivial-periodic – změna názvu kompilovaného kódu.

```
###### CONFIGURATION ######

### List of applications to be build
APPLICATIONS = client
```

- Úprava souboru Makefile pro trivial-periodic – přidání knihoven ke kompilaci:

```
###### SPECIAL TARGET RULES ######
rtprint: rtprint.c
$(CC) $(CFLAGS) $? $(LDFLAGS) -lrtdk -lm -lhudaqlib -o $@
```

tdma-api.c (RTnet Example):

- Úprava souboru Makefile pro trivial-periodic – změna názvu kompilovaného kódu.

```
###### CONFIGURATION ######

### List of applications to be build
APPLICATIONS = tdma-api
```

- Úprava souboru Makefile pro trivial-periodic – přidání knihoven ke kompilaci.

```
CFLAGS=$(shell $(XENOCONFIG) --xeno-cflags) $(MY_CFLAGS)
-I/usr/src/rtnet-0.9.12/stack/include

LDFLAGS=$(shell $(XENOCONFIG) --xeno-ldflags) $(MY_LDFLAGS) -lnative
-lrtdm # -lpthread -lxenomai  -lnative -lrtdk
```

rtt-sender.c a responder.c (RTnet Examples)

- Tyto zdrojové kódy, jako ostatní příklady ve složce *Examples/Xenomai/posix* se podařily zkompilovat bez použití skriptu pomocí příkazu ke kompilaci *gcc:*

```
gcc -c -Wall -ggdb -o rtt-sender.o -I/usr/xenomai/include -
I/usr/src/rtnet-0.9.12/stack/include rtt-sender.c
gcc -o rtt-sender -L/usr/xenomai/lib -lrt rtt-sender.o -lpthread
```

Příkazy kompilování ostatních příkladů obsažených v RTnet jsou součástí přílohy 3.

9 Zhodnocení dosažených výsledků.

9.1 Vlastnosti vybrané platformy pro řízení v reálném čase

Vybraná platforma stolního počítače v kombinaci s operačním systémem GNU/Linux 2.6.32 se ukázala s použitým RT rozšířením Xenomai 2.5.4 jako plně dostačující k řízení vybraného modelu Kulička na tyči od firmy Humusoft.

Výše zmíněný operační systém byl nainstalován na vybraný hardware stolního PC a Notebooku. Následně proběhly testy, kterými se ověřily základní parametry, které musí RTOS splňovat v závislosti na vykonávaných úlohách. Rozdíly mezi optimalizovaným Xenomai, nainstalovaným na PC a neoptimalizovaným na Notebooku, na kterém byla instalována verze pro PC, byly znatelné. Kvalitnějších výsledků dosahovalo méně výkonné PC, a to i když naměřené časy latencí byly v průměru vyšší než u výkonnějšího Notebooku s dvou-jádrovým procesorem. Žádaný determinismus byl u tohoto řešení v časové stabilitě, tu mělo PC větší. Perioda řízení, která byla stanovena simulací, ve vztahu ke stabilitě soustavy vybraného modelu na 20-100ms, je řádově více jak tisíckrát větší oproti časovému chvění (jitter), které se pohybovalo okolo 10µs u PC.

9.2 Vlastnosti použitých komunikací

Po neúspěšném pokusu zprovoznit deterministickou komunikaci protokolem RTnet, jak z důvodu náročnosti nastavování konfiguračních souboru, u kterých je nutné pro pochopení mechanizmu v něm, prostudovat celou dokumentaci, tak z důvodu nedostatečné podpory RT ovladačů sítových karet Ethernetu, které jsme ke komunikaci používali (82567LM Gigabit NET v PC a 82573V Gigabit NET v Notebooku). V případě náhrady karty v Notebooku se nenalezla vhodná karta do slotu ExpressCard/54.

Nakonec se podařilo nastavit konfigurační soubory tak, aby komunikace probíhala alespoň v rámci místní sítové smyčce (localhostu). Časové odezvy byly dle očekávání v řádech jednotek mikrosekund a dala by se použít na velice rychlou komunikaci např. pří použití virtuálního počítače nebo pří virtualizaci operačního systému.

Práce splnila koncepci NCS díky komunikaci s protokolem UDP, jejíž testy potvrdily možnost použití v oblastech Soft Real-time. Doby průměrné odezvy bez výrazného kolísání, kdy byl server schopen posílat a přijímat zprávy ohledně řízení soustavy, se pohybovaly s Notebookem v roli klienta (regulátoru) kolem 1ms, při nezatížené síti. V případě zatížené sítě

byly v mnoha případech časy odezvy i menší než při zatížené síti, ale špičkové odchylky byly v součtu až 13ms.

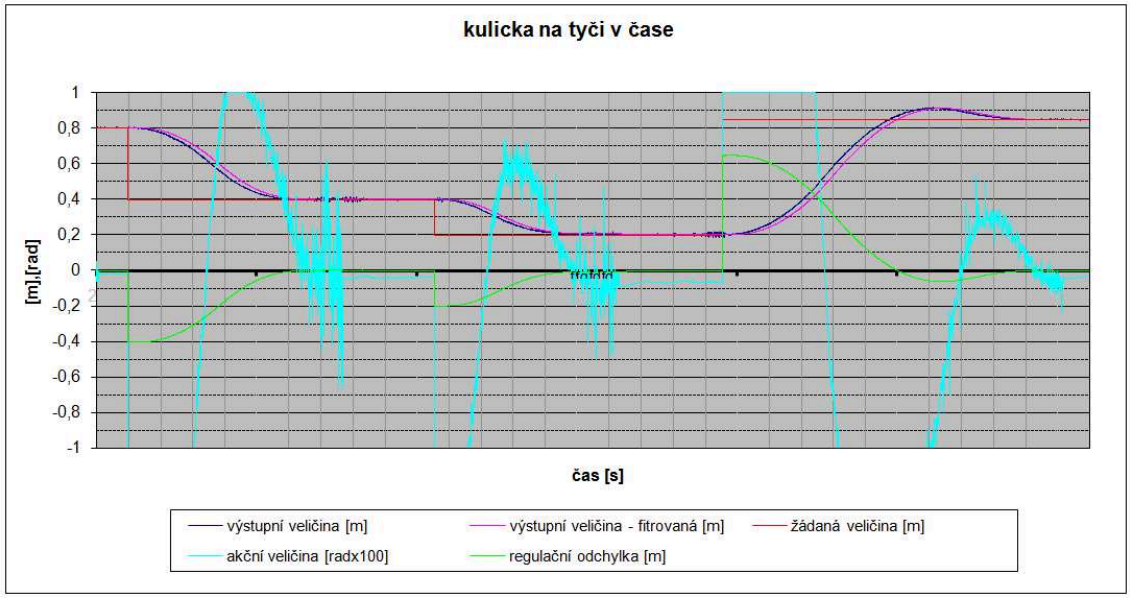
Na základě těchto testů by se pak mohla zvolit možná perioda řízení, v závislosti na to jak tvrdé podmínky by z hlediska Soft Real-time požadavků musil systém řízení splňovat, respektive kolik překročení stropového času (Deadline) za jednotku času a jak velké by bylo možné akceptovat. V našem případě použití periody řízení 10ms nemělo, vzhledem ke stabilitě soustavy a kvalitě regulace okem viditelný vliv oproti nesíťovému řízení.

9.3 Vlastnosti implementovaného regulátoru

Dle předpokladů průběhů regulace, které byly vzneseny simulací v simulačním prostředí MatLab / Simulink, byl implementován regulátor s podobným chováním na reálném modelu. Syntéza PID regulátoru proběhla ve výše zmíněném simulačním programu.

Byl použit PSD algoritmus odvozený z PID regulátoru, s Anti-Wind-Up doplňkem, který minimalizuje překmit na výstupu soustavy. Středová poloha na tyči byla dosažena z krajní pozice za 5s bez překmitu jak ukazuje graf 9.3.

Program je schopen na základě vyhodnocení parametrů regulační odchylky v čase změnit konstanty regulátoru (snížení derivační složky). Tato funkce se spolu s filtrací výstupní veličiny modelu ukázala jako nezbytná pro ustálení akční veličiny, která z důvodu přítomnosti šumu výstupní veličiny vyvolávala vysoké akční zásahy rychle se měnící v čase. Ve výsledku by se bez použití těchto opatření rameno s tyčí neustálilo a společně s kuličkou by nedosáhly úplné stabilizace.



Obr. 9.3 Průběhy různých veličin spojených s regulací při regulaci kuličky na tyči

10 Závěr

Závěrem lze říci, že byly splněny všechny důležité body, které plynou ze zadání diplomové práce. Podařilo se realizovat řízení, postavené na koncepci NCS, což znamená, že regulační smyčka byla uzavřená přes síť, v našem případě to bylo přes lokální Ethernetovou síť.

Pro řízení byl vybrán model kuličky na tyči od firmy Humusoft, který obsahuje nakloněnou rovinu, která lze řídit. Byly uskutečněny testy, na jejichž základě byl zvolen rozsah periody, ve kterém je vhodné zvolený model řídit, čímž jsme si také výsledky v simulačním prostředí MatLab / Simulink potvrdili.

Testy, které proběhly pro ověření dobrých Real-timových vlastností na operačním systému Linux s RT rozšířením Xenomai, byly srovnány mezi PC a Notbookem, na kterém byl Xenomai také nainstalován. Lepší RT vlastnosti nabídlo méně výkonné PC, protože na něj bylo nahráno optimalizované jádro OS, oproti notebooku.

Testy UDP komunikace ukázali vcelku dobrý determnismus při nezatížené síti, doba odezvy při testech mezi serverem a klientem dosahovala při nejlepších výsledcích průměrně 1063µs a jitter byl při tom 19µs. Při zatížené síti byly sice průměrné výsledky v mnoha případech i s menší odezvou než v případě nezatížené sítě, ale jitter stoupal v případě notebooku jako klienta se zatížením sítě. I když ping vytvořené UDP komunikace v čase špičkově přesahoval periodu řízení a akční zásah přicházel k modelu se zpožděním, na regulaci to nemělo okem pozorovatelné žádný vliv, systém řízení by tak jistě splňoval podmínky k zařazení do skupiny Soft Real-time.

11 Použitá literatura a zdroje:

[1] WANG, Fei-Yue; LIU, Derong. *Networked control systems*. 1. vyd. London: Springer-Verlag London Limited, 2008. ISBN 978-1-84800-214-2.

[2] ZEZULKA, František; HYNČICA, Ondřej. Průmyslový Ethernet IX: EtherNet/IP, EtherCAT. *AUTOMA* [online]. 2008, č. 10 [cit. 2012-02-17]. Dostupné z <http://www.odbornecasopisy.cz/index.php?id_document=37910>.

[3] ZEZULKA, František; HYNČICA, Ondřej. Průmyslový Ethernet IV: Principy průmyslového Ethernetu. *AUTOMA* [online]. 2007, č. 10 [cit. 2012-02-17]. Dostupné z <http://www.odbornecasopisy.cz/index.php?id_document=34198>.

[4] *Wikipedie – Ethernet* [online]. c2012 [cit. 2012-02-17]. Dostupné z <http://cs.wikipedia.org/wiki/Ethernet>.

[5] VOJČINÁK, Petr; PIEŠ, Martin; HÁJOVSKÝ, Radovan. Návrh LQG řízení pro fyzikální model kuličky na tyči. In Humusoft, s.r.o.. *Technical Computing Prague 2009 : Sborník příspěvků 17.ročníku konference.* Praha : Humusoft, s.r.o, 2009. 1x CD-ROM. Dostupné z <http://dsp.vscht.cz/konference_matlab/MATLAB09/prispevky/109_vojcinak.pdf>. ISBN 978-80-7080-733-0.

[6] MACHÁČEK, Zdeněk.; SROVNAL, Vilém. *Popis, identifikace systému a návrh regulátoru pomocí MATLAB v aplikaci Fotbal robotů*. Fakulta elektrotechniky a informatiky, VŠB-TU Ostrava [online]. c2006 [cit. 2012-02-18]. Dostupné z <http://dsp.vscht.cz/konference_matlab/MATLAB07/prispevky/machacek_srovnal/machacek_srovnal.pdf>.

[7] KISZKA, Jan. *RTnet – Documentation* [online]. c2002-2009 [cit. 2012-02-18]. Dostupné z <http://www.rtnet.org/doc.html>.

[8] *Xenomai, RTnet: Programming* [online]. c2007 [cit. 2012-02-18]. Dostupné z <http://www.xenomai.org/index.php/RTnet:Programming>.

[9] PELČÁK, Vít. *ABC Linuxu – Realtime modifikace Linuxu – 1 (RTLinux a RTAI)* [online]. c2006 [cit. 2012-02-19]. Dostupné z <http://www.abclinuxu.cz/clanky/system/real-time-modifikace-linuxu-1-rtlinux-a-rtai>.

[10] *HUMUSOFT – Měřící karty* [online]. c1991-2012 [cit. 2012-02-19]. Dostupné z <http://www.humusoft.cz/produkty/datacq/>.

[11] GERUM, Philippe. *RTAI/fusion real-time extension* [online]. c2004 [cit. 2012-02-19]. Dostupné z <http://lkml.indiana.edu/hypermail/linux/kernel/0410.1/0844.html>.

[12] KRČMÁŘ, Petr. *Roots.cz - Historie operačního systému GNU/Linux* [online]. c2010 [cit. 2012-02-19]. Dostupné z <http://www.root.cz/texty/historie-operacniho-systemu-gnulinux/>.

[13] *Wikipedie – Linux* [online]. c2012 [cit. 2012-02-20]. Dostupné z <http://cs.wikipedia.org/wiki/Linux>.

[14] KRČMÁŘ, Petr. *Roots.cz - Jádro 2.6.32 nekončí, vývoj ale bude pomalejší* [online]. c2012 [cit. 2012-02-20]. Dostupné z <http://www.root.cz/zpravicky/jadro-2-6-32-nekonci-vyvoj-ale-bude-pomalejsi/>.

[15] KISZKA, Jan; WAGNER, Bernardo; ZHANG, Yuchen; BROENINK, Jan. *A Flexible Hard Real-Time Networking Framework.* University of Hannover, Germany; University of Twente, the Netherlands [online]. c2005 [cit. 2012-04-28]. Dostupné z <http://www.rtnet.org/download/RTnet-ETFA05.pdf>.

[16] KISZKA, Jan; SCHWEBEL, Robert. *Alternative: Rtnet* [online]. c2004 [cit. 2012-04-28]. Dostupné z <http://www.rtnet.org/download/ad104705.pdf>.

[17] *Wikipedie – Operační systém reálného času* [online]. c2011 [cit. 2012-04-29]. Dostupné z <http://cs.wikipedia.org/wiki/Opera%C4%8Dn%C3%AD_syst%C3%A9m_re%C3%A1ln%C3%A9ho_%C4%8Dasu>.

[18] *Wikipedie - Operační systém reálného času, základní charakteristika* [online]. c2011 [cit. 2012-04-29]. Dostupné z <http://cs.wikipedia.org/wiki/Opera%C4%8Dn%C3%AD_syst%C3%A9m_re%C3%A1ln%C3%A9ho_%C4%8Dasu#Z.C3.A1kladn.C3.AD_charakteristika_RTOS>.

[19] KAČMÁŘ, Dalibor. *Microsoft Windows CE verze 2.12 a 3.0 a reálný čas*. *AUTOMA* [online]. 2001, č. 01 [cit. 2012-02-30]. Dostupné z <http://www.odbornecasopisy.cz/index.php?id_document=33433>.

[20] REISNER, Ladislav. Reálný čas ve Windows s použitím RTX. *AUTOMA* [online]. 2001, č. 09 [cit. 2012-02-30]. Dostupné z <http://www.odbornecasopisy.cz/index.php?id_document=33664>.

[21] DOSTÁL, Radim. *Builder.cz - Protokol UDP 1.část* [online]. c2012 [cit. 2012-05-01]. Dostupné z <http://www.builder.cz/art/cpp/udp1.html>.

[22] *Xenomai - File:Xenomai co-kernel.png* [online]. c2009 [cit. 2012-05-01]. Dostupné z <http://www.xenomai.org/index.php/File:Xenomai_co-kernel.png>.

[23] WINKLER, Zbyněk. *robotika.cz - Měření rychlosti* [online]. c2005 [cit. 2012-05-02]. Dostupné z <http://robotika.cz/guide/filtering/en>.

[24] *Číslicové řízení, regulátor PSD.* Ústav mechatroniky a technické informatiky, Technická univerzita v Liberci [online]. c2000 [cit. 2012-05-02]. Dostupné z <http://www.mti.tul.cz/files/zsr/cislicove-rizeni-PSD-regulator.pdf>.

[25] RACCIU, Giovanni; MANTEGAZZA, Paolo. *RTAI 3.4 User Manualrev 0.3* [online]. c2006 [cit. 2012-05-03]. Dostupné z <http://www.scribd.com/doc/77113792/RTAI-User-Manual-34-03>.

[26] STRNADEL, Josef. Návrh časově kritických systémů I: specifikace a verifikace. *AUTOMA* [online]. 2010, č. 10 [cit. 2012-05-03]. Dostupné z <http://www.odbornecasopisy.cz/index.php?id_document=42105>.

12 Přílohy

Příloha 1: vytvoření PID regulátoru v MatLabu

```
G=tf([7.1428], [1 0 0]);
syms t s;
Gs = (7.1428)/(s^2);
H = 1/s*Gs;              %obraz prech. char
Gsh = ilaplace(H);       %prechodova charakteristika
figure
ezplot(Gsh, [0 10]);
grid on;
axis auto;
nastaveni=pidtuneOptions('CrossoverFrequency',6.35,'PhaseMargin',85);
[Reg info]=pidtune(G,'pid',nastaveni)
R=tf(Reg);
H=feedback(R*G,1)
step(H)
```

Příloha 2: vytvoření PSD regulátoru v MatLabu

```
Kp=0.49233;
Ki=0.024776;
Kd=0.38937;
T=0.075;
Ps=tf(Kp)
Is=tf([Ki],[1 0])
Ds=tf([Kd 0],[1])
display('Rr=')
Rr=Ps+Is+Ds
display('OL=')
OL=feedback(Rr*G,0)
[numOL,denOL]=tfdata(OL);
display('CL=')
CL=feedback(Rr*G,1)
[numCL,denCL]=tfdata(CL);
figure(2)
step(CL)           %odezva uzavrene smycky
grid on;
hold on;
Hd=c2d(CL,T,'zoh') %diskretizace
step(Hd)
hold off;
%------------vypocet PSD-----------------
b0=Kp+(Kd/T);
b1=Ki*T-Kp-((2*Kd)/T);
b2=Kd/T;
```

Příloha 3: kompilace příkladů RTnet

rtt-sender a rtt-responder:

```
gcc -c -Wall -ggdb -o rtt-sender.o -I/usr/xenomai/include -I/usr/src/rtnet-
0.9.12/stack/include rtt-sender.c
gcc -o rtt-sender -L/usr/xenomai/lib -lrt rtt-sender.o -lpthread
```

rttcp-client a rttcp-server:

```
gcc -c -Wall -ggdb -o rttcp-client.o -I/usr/xenomai/include -
I/usr/src/rtnet-0.9.12/stack/include rttcp-client.c
gcc -o rttcp-client -L/usr/xenomai/lib rttcp-client.o -lpthread -lrt
```

eth_p_all:

```
gcc -c -Wall -ggdb -o eth_p_all.o -I/usr/xenomai/include -I/usr/src/rtnet-
0.9.12/stack/include eth_p_all.c
gcc -o eth_p_all -L/usr/xenomai/lib eth_p_all.o -lpthread -lrt
```

raw-ethernet:

```
gcc -c -Wall -ggdb -o raw-ethernet.o -I/usr/xenomai/include -
I/usr/src/rtnet-0.9.12/stack/include raw-ethernet.c
gcc -o raw-ethernet -L/usr/xenomai/lib raw-ethernet.o
```

Příloha 4: (Na CD) Zdrojové kódy

Příloha 5: (Na CD) Grafy z testování komunikace a průběhy regulace

Příloha 6: (Na CD) Soubory MatLab / Simulink (*mdl, m*)